PCT

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : **G09G 3/02**	**A1**	(11) International Publication Number: **WO 95/21436**
		(43) International Publication Date: 10 August 1995 (10.08.95)

(21) International Application Number: PCT/US95/01483

(22) International Filing Date: 3 February 1995 (03.02.95)

(30) Priority Data:
108,565   4 February 1994 (04.02.94)   IL

(71) Applicant *(for all designated States except US)*: BARON MO-TION COMMUNICATIONS, INC. [US/US]; Mr Michael Adar, 706 Chimelus Drive, Palo Alto, CA 94306 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: BARON, Ehud [IL/IL]; 61 Haag Street, 34980 Haifa (IL). GENOSSAR, Omry [IL/IL]; 4 Tadhar Street, 36803 Haifa (IL). PRISHVIN, Alexander [IL/IL]; 82/30 Hantke Street, 96629 Jerusalem (IL).

(74) Agents: GALLOWAY, Peter, D.; Ladas & Parry, 26 West 61st Street, New York, NY 10023 (US) et al.

(81) Designated States: AM, AT, AU, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, EE, ES, FI, GB, GE, HU, JP, KE, KG, KP, KR, KZ, LK, LR, LT, LU, LV, MD, MG, MN, MW, MX, NL, NO, NZ, PL, PT, RO, RU, SD, SE, SI, SK, TJ, TT, UA, US, UZ, VN, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG), ARIPO patent (KE, MW, SD, SZ).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: IMPROVED INFORMATION INPUT APPARATUS

(57) Abstract

Information input apparatus (10, 40, 50, 60, 70) including body supported apparatus (10) for sensing voluntary body motions and providing an output indication thereof, a symbol output interpreter (140) operative to utilize the output indication for providing symbol outputs, and a motion output interpreter (140) operative to utilize the output indication for providing motion control outputs.

FIELD OF THE INVENTION

The present invention relates to input devices for computers generally.

BACKGROUND OF THE INVENTION

U.S. Patent 4,787,051 to Olson describes an inertial mouse system which uses three pairs of accelerometers.

U.S. Patent 4,839,836 to LaBiche et. al. describes an apparatus for providing spatial orientation data signals, using an inertial platform accelerometer cluster having a plurality of accelerometers.

U.S. Patent 5,181,181 to Glynn describes a computer apparatus input device for three-dimensional information.

SUBSTITUTE SHEET (RULE 26)

2

SUMMARY OF THE INVENTION

The present invention seeks to provide an improved input device.

There is thus provided in accordance with a preferred embodiment of the present invention information input apparatus including body supported apparatus for sensing voluntary body motions and providing an output indication thereof, a symbol output interpreter operative to utilize the output indication for providing symbol outputs, and a motion output interpreter operative to utilize the output indication for providing motion control outputs.

Further in accordance with a preferred embodiment of the present invention, the output indication represents features of body motion including features which are characteristic of the individual.

Still further in accordance with a preferred embodiment of the present invention, a mode selector is provided which is operative to cause a selected one of the symbol output interpreter and the motion output interpreter to function.

Further in accordance with a preferred embodiment of the present invention, the body supported apparatus is a hand held device.

Still further in accordance with a preferred embodiment of the present invention, the body supported apparatus is a generally pen-shaped device.

Additionally in accordance with a preferred embodiment of the present invention, the generally pen-shaped device is operative to provide a visible writing function.

Still further in accordance with a preferred embodiment of the present invention, the information input apparatus also includes an object whose motion is

SUBSTITUTE SHEET (RULE 26)

3

controlled by the motion control outputs.

Further in accordance with a preferred embodiment of the present invention, the object is a graphic object displayed on a display or a physical object.

Further in accordance with a preferred embodiment of the present invention, the symbol outputs represent alphanumeric symbols or a sensory quality such as an acoustic stimulus, including but not limited to music, or such as a visual stimulus, including but not limited to a color or a color image.

It is appreciated that the applicability of the present invention is very broad and is suitable for the following fields of use, inter alia: games such as video games, toys, model vehicles, robotics, simulations such as flight simulations, technical drawings and CAD.

Still further in accordance with a preferred embodiment of the present invention, the information input apparatus also includes a computer, having a location input and a symbol input, and a display operated by the computer and wherein the symbol outputs represent information to be displayed on the display and the motion outputs are supplied to the location input and are employed by the computer to govern the location of the information on the display.

Further in accordance with a preferred embodiment of the present invention, the symbol outputs include function commands.

There is also provided, in accordance with a preferred embodiment of the present invention, a method by which a manipulable device provides an output indication representing its own angular motion, the method including recording actual acceleration data from a plurality of accelerometers mounted in the manipulable device, generating predicted acceleration data on the basis of hypothetical angular motion information, comparing the predicted acceleration data to the actual accel-

SUBSTITUTE SHEET (RULE 26)

4

eration data, computing improved hypothetical angular motion information, repeating, while the predicted acceleration data differs significantly from the actual acceleration data, the generating, comparing and computing steps, and providing an output indication of the improved hypothetical angular motion information.

Further in accordance with a preferred embodiment of the present invention, the angular motion information includes angular displacement information, angular velocity information and angular acceleration information.

Still further in accordance with a preferred embodiment of the present invention, the method includes computing linear motion information from the improved hypothetical angular motion information and from the actual acceleration data.

Additionally in accordance with a preferred embodiment of the present invention, recording includes recording from at least four accelerometers mounted in the manipulable device, wherein the accelerometers each have a center of mass and wherein the centers of mass do not lie within a single plane.

Still further in accordance with a preferred embodiment of the present invention, the method also includes receiving the output indication of the improved hypothetical angular motion information and manipulating an object in accordance therewith.

There is additionally provided, in accordance with a preferred embodiment of the present invention, an accelerometer array mounted in a manipulable device and including at least four accelerometers each having a center of mass, wherein the centers of mass do not lie within a single plane, and a manipulable device motion computer receiving input from the accelerometers and generating an output signal indicative of the motion of the manipulable device.

SUBSTITUTE SHEET (RULE 26)

5

Further in accordance with a preferred embodiment of the present invention, the manipulable device motion computer is operative to:

record actual acceleration data from the accelerometers,

generate predicted acceleration data on the basis of hypothetical angular motion information,

compare the predicted acceleration data to the actual acceleration data,

compute improved hypothetical angular motion information,

repeat, while the predicted acceleration data differs significantly from the actual acceleration data, the generating, comparing and computing steps, and

provide an output indication of the improved hypothetical angular motion information.

Further in accordance with a preferred embodiment of the present invention, the apparatus also including an object manipulator receiving the output signal indicative of the motion of the manipulable device and manipulating an object in accordance therewith.

There is also provided, in accordance with a preferred embodiment of the present invention, an information input method including sensing voluntary body motions and providing an output indication thereof, utilizing the output indication for providing symbol outputs, and utilizing the output indication for providing motion control outputs.

SUBSTITUTE SHEET (RULE 26)

6

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will be understood and appreciated from the following detailed description, taken in conjunction with the drawings in which:

Fig. 1 is a simplified pictorial illustration of object control and handwriting recognition apparatus constructed and operative in accordance with a preferred embodiment of the present invention;

Figs. 2A and 2B are schematic drawings of preferred structures of portions of the apparatus of Fig. 1;

Fig. 3 is a simplified block diagram of the apparatus of Fig. 1;

Fig. 4 is a simplified flow chart illustrating the object control process performed by the apparatus of Fig. 1;

Fig. 5A is a simplified flow chart illustrating the process of step 440 of figure 4;

Fig. 5B comprises mathematical equations illustrating the process performed by Fig. 5A;

Fig. 6 is a simplified block diagram of the apparatus of Fig. 1;

Fig. 7A is a simplified flow chart illustrating the teaching process performed by the apparatus of Fig. 1;

Fig. 7B is a simplified flow chart illustrating the recognition process performed by the apparatus of Fig. 1; and

Figs. 8A and 8B are graphical illustrations useful in understanding a preferred method for a portion of the teaching and recognition processes performed by the apparatus of Fig. 1.

Appendix A is a computer listing of a preferred software implementation of a portion of steps 720 of Fig. 7A and 800 of Fig. 7B.

SUBSTITUTE SHEET (RULE 26)

7

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

Reference is now made to Fig. 1 which is a simplified pictorial illustration of apparatus operative to perform motion control in synergistic combination with symbol interpretation such as handwriting recognition. A hand-held pen 10 is operative to be translated and rotated about some or all of three perpendicular axes. The term "six degrees of freedom" is used herein to designate translation along and rotation about three orthogonal axes.

Pen 10 also comprises a plurality of built-in accelerometers 25, such as model ICS 3031-2 commercially available from IC Sensors, 1701 McCarthy Blvd., Milpitas, CA 95035. Preferably, pen 10 comprises six accelerometers arranged in pairs, with each pair lying along a particular axis, with the axes being mutually orthogonal. Alternatively, the axes may not be mutually orthogonal. In any case the accelerometers need not be coplanar.

Pen 10 also comprises a plurality of amplifiers 30, associated with the plurality of accelerometers 25. Fig. 2A is a schematic drawing of a preferred embodiment of amplifier 30.

Alternatively, removable apparatus comprising a plurality of accelerometers 25 as described above, and also comprising associated amplifiers 30, as described above, may be retrofitted onto the pen 10. The removable apparatus may have the form of a cap fitting the end of the pen, a ring fitting over the pen, or any other suitable form.

In a still further alternative, the apparatus may not include a pen, but may have any other suitable hand held form. In a yet further alternative, the apparatus may be in the form of a ring fitting the user's finger, may be supported by the body of a user, or mounted thereupon in any suitable matter.

SUBSTITUTE SHEET (RULE 26)

8

Pen 10 also comprises a switch 35, which can be used to send a signal indicating whether pen 10 is being used for handwriting recognition or as a pointing and control device. Alternatively the signal may be sent by moving pen 10 in a predefined format, or by any other appropriate means. During handwriting recognition, the user may write with pen 10 on writing surface 37.

The data from the plurality of accelerometers 25 in pen 10 is termed herein "accelerometer data". The accelerometer data is sent through a cable to a control circuit 40. Alternatively, the accelerometer data may be sent through any suitable wireless communication link, such as ultrasonic, infrared, or by any other suitable means.

Control circuit 40 amplifies the acceleration signals from pen 10 and converts them to digital form, preferably using an analog to digital converter. Fig. 2B is a schematic drawing of a preferred embodiment of an analog to digital converter suitable for the present application.

Control circuit 40 then sends acceleration data to a CPU 50. CPU 50 may be any suitable CPU such as an IBM PC compatible computer with an 80386 processor chip.

Associated with CPU 50 are a screen 60 and a keyboard 70. An object 80, such as a cursor or a graphic representation of a physical object, is displayed on screen 60. When pen 10 is used for object control, CPU 50, based on the acceleration data, moves the cursor or graphic representation 80 with six degrees of freedom, corresponding to the movement of pen 10.

A symbol 85, such as one or more characters or words, may also be displayed on screen 60. When pen 10 is used for handwriting recognition, CPU 50, based on the acceleration data, displays the symbols corresponding to what is written on writing surface 37 on screen 60.

The functionality of the apparatus of Fig. 1

SUBSTITUTE SHEET (RULE 26)

will now be described. Using switch 35 the user indi-
cates whether handwriting recognition or object control
is to be performed. Depending on the user's choice, the
apparatus of Fig. 1 performs the appropriate function.

The functionality of the apparatus of Fig. 1
when performing object control will now be briefly de-
scribed. The user moves pen 10 in three dimensions; the
motion may include six degrees of freedom. Pen 10 sends
acceleration data describing the accelerations of pen 10
during the motion to control circuit 40.

Control circuit 40 amplifies and digitizes the
acceleration data. The data is sent by control box 40 to
CPU 50.

CPU 50 computes the translational displacement,
velocity, and acceleration of pen 10 along three mutually
perpendicular axes which axes need have no relation to
the axes of the accelerometers. CPU 50 also computes the
angular displacement (rotation), velocity and accelera-
tion of pen 10 around the same three mutually perpendic-
ular axes.

Based on the computed output, CPU 50 moves the
cursor or the representation of an object 80 on screen 60
with translations and rotations corresponding to those of
pen 10. The axes for the translation and rotation of the
cursor or object correspond to the axes used to compute
the translation and rotation of pen 10.

Reference is now additionally made to Fig. 3
which is a simplified block diagram of the apparatus of
Fig. 1. Pen 10, when moved by the user with six degrees
of freedom, transmits data describing the accelerations
of pen 10 to amplification circuit 120. Amplification
circuit 120 amplifies the acceleration data and transmits
the amplified acceleration data to analog/digital con-
verter 130. Analog/digital converter 130 digitizes the
acceleration data and transmits the digitized data to
displacement/velocity/acceleration computation apparatus

SUBSTITUTE SHEET (RULE 26)

10

140, termed herein DVA 140.

DVA 140 computes the angular displacement, velocity, and acceleration of pen 10 around three mutually perpendicular axes which axes need have no relation to the axes of the accelerometers. DVA 140 also computes the translational displacement, velocity and acceleration of pen 10 along the same three mutually perpendicular axes.

DVA 140 transmits data describing the six degrees of freedom to screen display control 150. Based on the data, screen display control 150 updates screen 60 to show the new location and orientation of the cursor or the other object depicted on screen 60.

Reference is now additionally made to Fig. 4 which is a simplified flow chart illustrating operation of the apparatus of Fig. 1 in accordance with a preferred embodiment of the invention. The preferred method of operation of the method of Fig. 4 includes the following steps:

STEP 410: Read accelerometer data. Data from each of the plurality of accelerometers 25 is sampled, preferably at a rate of one thousand data points per second.

STEP 412: Check whether session is at the beginning. At the beginning of a session, STEP 420, described below, is required.

STEP 415: Check whether pen is in motion. The accelerometer data is analyzed to determine whether pen 10 is in motion.

Preferably, pen 10 is considered to be not in motion whenever all of the acceleration signals indicate that the only sensed accelerations are due to gravity. Signals are chosen from one member of each of three pairs of accelerometers, each pair arranged along a different axis.

Let the vector $U=(U_1,U_2,U_3)$ denote the signals

SUBSTITUTE SHEET (RULE 26)

11

of the three accelerometers. Let the matrix $A=(K_1, K_2, K_3)$ denote the sensitivities of each of the three accelerometers.

The sensitivities of the accelerometers correct for any deviations of the axes of each pair of accelerometers as they are actually mounted in pen 10 from the common axis on which they are supposed to be situated; this component of the sensitivity is called static sensitivity. The sensitivities also correct for deviations between the axes of the global orthogonal coordinate system and the axes of the pairs of accelerometers; this component of the sensitivity is called dynamic sensitivity. In actual practice, both static sensitivity and dynamic sensitivity may make important contributions to sensitivity.

The static sensitivity is computed as part of step 420, described in detail below. The dynamic sensitivity is computed as part of step 455, described in detail below.

Let ϵ denote a small positive constant; for example, .005g where g denotes the acceleration of gravity at the earth's surface. Then the pen is considered not to be in motion whenever $1-\epsilon < |A^{-1}U| < 1+\epsilon$.

STEP 420: Compute initial conditions. The initial conditions may comprise the initial Euler angles between the global coordinate system and the axes of the pairs of accelerometers. These Euler angles are now determined.

Assuming that, at the initial condition, either at the beginning of the session or when the pen is not in motion, the only accelerations measured are due to gravity, the static sensitivity can be computed from the accelerometer data while the pen is at rest in three known orientations. Alternatively, the static sensitivity can be computed once as a property of the pen and stored for future use.

SUBSTITUTE SHEET (RULE 26)

12

STEP 430: Compute the differential signal from each pair of accelerometers. The signals from each member of each pair of accelerometers are subtracted to form a differential signal for each pair of accelerometers.

STEP 440: Compute rotational parameters. The rotational parameters define parameters of the motion about the three axes of the global coordinate system. The rotational parameters comprise the three Euler angles; the three angular velocities; and the three angular accelerations.

The rotational parameters are computed in parallel using an iterative feedback loop. In each iteration of the loop, an estimated differential acceleration is computed from the current rotational parameters. If the difference between the estimated differential acceleration and the actual differential acceleration signal is less than a predetermined amount, iteration is terminated.

Otherwise, new values of the parameters are estimated in each iteration from the previous values and the difference between the estimated differential acceleration and the actual differential acceleration data. The method of this step is described more fully below with reference to Fig. 5.

STEP 450: Compute translation acceleration. The angular orientation and the angular acceleration are known from step 440. From the angular orientation and the sensitivity vector the acceleration due to gravity is computed.

Given the angular acceleration and the acceleration due to gravity as well as the sensitivity vector, the translational acceleration is computed. The computation is according to the formula $a_t = K^{-1}u - a_g - a_r$ where a_t is the translational acceleration; K is the sensitivity vector; u is the signal of one accelerometer; a_g is the

SUBSTITUTE SHEET (RULE 26)

13

component of the acceleration of gravity sensed by the
one accelerometer; and a_r is the angular acceleration.

STEP 455: Update dynamic sensitivity. As
explained above, the dynamic sensitivity represents
deviations between the axes of the global orthogonal
coördinate system and the axes of the pairs of accelerom-
eters. Since the angular orientation of pen 10 may have
changed, the dynamic sensitivity may also have changed.

Given the change in the angular orientation of
pen 10, the new dynamic sensitivity may be computed from
the new angular orientation and the old matrix of dynamic
sensitivity.

STEP 460: Compute translational velocity and
displacement. The translational velocity is computed by
integrating the translational acceleration with respect
to time. The displacement is computed by integrating the
translational velocity with respect to time.

STEP 470: Move screen object. Based on the
output of previous steps which comprises translational
acceleration, velocity and displacement as well as angu-
lar acceleration, velocity and orientation, the screen
object is moved. The moving of the screen object may be
according to any appropriate transformation of the mo-
tions of pen 10.

Reference is now additionally made to Fig. 5A
which is a simplified flowchart illustrating to operation
of step 440 of Fig. 4. Fig. 5A includes the following
steps:

STEP 480: Set initial parameters. The rota-
tional parameters comprise the three Euler angles; the
three angular velocities; and the three angular accelera-
tions. The initial value for the Euler angles is comput-
ed based on the previously known value of the parameters,
assuming that the acceleration has remained constant.

STEP 482: Compute differential acceleration
from model. First the position of an accelerometer in

SUBSTITUTE SHEET (RULE 26)

the coordinate system of the pen is defined by vector r and the rotation of the pen in the global coordinate system is defined by a rotation matrix A(phi). For example, A(phi) may be an appropriate rotation matrix as presented in sections 14.10-5 through 14.10-7, pages 475-480 of Mathematical Handbook for Scientists and Engineers by Korn and Korn, 2nd Edition, published by McGraw-Hill in 1968. Here phi is a vector of Euler angles: phi=$(\alpha,\beta,\Gamma)^T$. Then the position of the accelerometer in the global coordinate system R is defined by R=Ar.

Reference is hereby additionally made to Fig. 5B, which contains equations useful for understanding the steps of Fig. 5A. Equation 490 illustrates the computation of the acceleration of the accelerometer in the global coordinate system.

As the Euler angles of the accelerometer change, the sensitivity vector K also changes. The change in the sensitivity vector K may be computed by using equation 492 of Fig. 5B.

Given the acceleration of the accelerometer and the new sensitivity vector, the estimated value for the differential signal of the accelerometer, u_{est} may be computed by using equation 494 of Fig. 5B.

The remainder of the parameters may be computed with an appropriate model. Preferably, a model which allows the use of only the parameters specified above, rather than a larger number of parameters, is used. For example, equation 496 of Fig. 5B represents an appropriate model for computing the remainder of the parameters.

STEP 484: Is the difference between the computed and the current value less than a predetermined amount? If the difference is less than this amount, the estimated parameters are taken to be correct and iteration is terminated, with the computed parameters being reported.

An appropriate value for the predetermined

SUBSTITUTE SHEET (RULE 26)

15

amount may vary depending on, for example, the maximum
number of desired iterations. One possible appropriate
value would be .0003 g, where g represents the accelera-
tion of gravity at the earth's surface.

STEP 486: Compute changes in estimated angles
according to the gradient method. New estimated angles
are computed by adding a change to the old estimated
angles; the change is computed according to the gradient
method. The gradient method is explained more fully in
section 20.3-3 of Mathematical Handbook for Scientists
and Engineers by Korn and Korn, referred to above.

STEP 488: Compute new parameters. New values
for the remaining parameters are computed. Iteration
then continues with step 482.

The functionality of the apparatus of Fig. 1
when performing handwriting recognition will now be
briefly described. Pen 10 sends acceleration data
through control circuit 40 to CPU 50. Teaching and
recognition then occur based on the data from pen 10.

Reference is now additionally made to Fig. 6
which is a simplified block diagram of the apparatus of
Fig. 1 when used for handwriting recognition. The appa-
ratus of Fig. 6 receives input from pen 10.

Pen 10, when moved by the user of the handwrit-
ing recognition apparatus, transmits data describing the
accelerations of pen 10 over time to acceleration teach-
ing control 630 and/or acceleration handwriting recogni-
tion control 650.

The data from pen 10 may be transmitted to
acceleration teaching control 630. Transmission to accel-
eration teaching control 630 typically occurs for each
person who is to use the system for handwriting recogni-
tion for the first time. Transmission to acceleration
teaching control 630 also preferebly occurs when recogni-
tion errors are detected; use of acceleration teaching
control 630 when recognition errors are detected is

SUBSTITUTE SHEET (RULE 26)

16

termed herein adaptive teaching.

Acceleration teaching control 630 operates on the data received, which data represents hand movements by the user when writing a symbol, together with manually-provided identification of the symbol codes that are associated with the data. Acceleration teaching control 630 then updates database 640, a per-person per-symbol acceleration database. Database 640 comprises prototypes of accelerations for each symbol, comprising data specific to each person for each symbol.

Alternatively, the data from pen 10 may be transmitted to acceleration handwriting recognition control 650. Acceleration handwriting recognition control 650 operates on the data received from pen 10 to recognize the symbol represented by the movement of pen 10.

The output of acceleration handwriting recognition control 650 comprises a list of symbol codes and their respective probabilities. An acceleration handwriting recognition post-processing circuit 660, chooses the correct symbol code based on the list of symbol codes and probabilities, and on post-processing information which preferably comprises a database of previous confusions and a dictionary. The output of acceleration handwriting recognition post-processing circuit 660 is a list of symbol codes and/or words sorted by likelihood.

Reference is now additionally made to Figs. 7A and 7B which are simplified flow charts illustrating operation of the apparatus of Fig. 7 in accordance with a preferred embodiment of the invention, when performing handwriting recognition. Fig. 7A illustrates the teaching process and Fig. 7B illustrates the recognition process. The steps in Fig. 7A include the following:

STEP 710: Read accelerometer data. The accelerometer data comprises data points representing sampling of the acceleration measured by accelerometers 25.

SUBSTITUTE SHEET (RULE 26)

17

Preferably, the sampling rate is approximately 1600 data
points per second, averaged over 8 points, producing an
output of approximately 200 data points per second.

STEP 712: Identify pen-surface contact termi-
nation.   The data from step 710 does not include the
surface contact status of pen 10. The surface contact
status of pen 10 may be derived from the acceleration
data as follows:

The acceleration data is filtered to remove
components other than noise. For example, the accelera-
tion data may be filtered by a Butterworth digital filter
described in Digital Filter Design by T.W. Parks and C.S
Burrus, John Wiley & Sons, 1987, chapter 7, section
7.3.3, using the 4th order lowpass digital filter with a
cut-off frequency of 0.7 to 0.9.

The filtered acceleration data is then inte-
grated over time. The slope of the integrated filtered
acceleration data is then analyzed to determine the point
at which the slope exceeds a threshold value. The point
at which the slope exceeds the threshold value is taken
to be the first point with status "pen down". The point
at which the slope falls below a threshold value is taken
to be the first point with status "pen up"; the threshold
value may or may not be the same as the previously de-
scribed threshold value.

The threshold values described above may be
determined in advance for the particular type of pen and
writing surface, may be determined by a learning process
for the particular person, or may be determined by other
means.

STEP 715:   Identify individual symbols and
words.   The data from the previous step is divided into
data representing individual symbols. The status which
comprises the status of "pen up" is termed herein "pen
not down".   Preferably, the number of consecutive data
points with status of "pen not down", which data points

SUBSTITUTE SHEET (RULE 26)

18

represent a particular duration of the status "pen not
down" is taken to indicate the end of a symbol or of a
word.

Typically, the duration of status "pen not
down" within a range from 200 milliseconds to 400 milli-
seconds is taken to indicate the end of a symbol.   Dura-
tion of the status "pen not down" in the range from 800
milliseconds to 1200 milliseconds is typically taken to
indicate the end of a word.

Alternatively, the end of a symbol or of a word
may be indicated by data points which represent pen
movements that are not part of a symbol, or by other
means.   Output data from step 715 comprises symbol end
and word end data.

STEP 720: Normalize accelerometer data.   The
accelerometer data is normalized in time or by other
means.   Appendix A is a computer listing in the C pro-
graming language comprising routines that are a preferred
implementation of step 720. The routines comprise the
following routines in section II, "pre-preprocessing":
normal; together with various definitions used by routine
normal.

STEP 730:   Filter accelerometer data.   The
normalized accelerometer data received from the previous
step is filtered in order to remove noise. The filtering
may be accomplished by iterative smoothing of adjacent
points until the total change in the signal due to a
smoothing operation is less than the desired accuracy of
the data, or by other suitable means.

STEP 740:   Parameterize accelerometer data.
The data is parameterized according to criteria which are
chosen to represent each symbol. If the accelerometers
are not mutually orthogonal, the acceleration data may be
converted into equivalent data in a mutually orthogonal
coordinate system as follows:

Let the non-orthogonal signals be denoted by

SUBSTITUTE SHEET (RULE 26)

19

the vector $u=(u_1,u_2,u_3)^T$ and the orthogonal signals be denoted by the vector $u'=(u'_1,u'_2,u'_3)^T$. Then $u'=A_0A^{-1}u$ where A is a vector of static sensitivity vectors $A=(A_1,A_2,A_3)$ of the three accelerometers. The static sensitivity vector is computed from the outputs of the accelerometers during a defined orientation without movement. A_0 is a diagonalized matrix of sensitivity of the orthogonal coordinate system comprising the norms of A_1, A_2, and A_3.

The parameters preferably comprise the following:

number of points before normalization;

normalized signal of pen status;

normalized signal of Z acceleration;

sine of the angle α' which angle is defined as the angle between the vector associated with the current data point $(AccX_i,AccY_i,AccZ_i)$ and the AccXAccY plane as shown in Fig. 8A;

cosine of the angle α';

sine of the angle β' which angle is defined as the angle between the vector that connects the point before the previous data point $(AccX_{i-2},AccY_{i-2},AccZ_{i-2})$ and the current point $(AccX_i,AccY_i,AccZ_i)$, and the vector that connects the current point with the point after the subsequent point $(AccX_{i+2},AccY_{i+2},AccZ_{i+2})$ in space $(AccX,AccY,AccZ)$ as shown in Fig. 8B;

and cosine of the angle β'.

STEP 750: Generalize parameters. The parameters of the symbol being learned represent a specific instance of the symbol. The symbol prototype stored by the system is to represent the general characteristics of the symbol as drawn by that person. Therefore, the parameters of the symbol being learned are generalized by some suitable means, such as by computation of the average of the value of each parameter from previous instances of the symbol along with the value of each param-

SUBSTITUTE SHEET (RULE 26)

20

eter from the current instance of the symbol.

STEP 760: Update per-person per-symbol accel-
eration prototype database. The newly computed parame-
ters from the previous step are stored in the per-person
per-symbol acceleration prototype database.

The steps in Fig. 7B include steps which have
already been described above with reference to Fig. 7A.
The remainder of the steps in Fig. 7B include the follow-
ing:

STEP 800: For each prototype in the per-person
per-symbol acceleration prototype database, build a
measure of comparison between the sample and the proto-
type, combined over parameters in the prototype. In
accordance with a preferred embodiment of the present
invention, all parameters are combined together to pro-
duce the measure of comparison. Appendix A is a computer
listing in the C programing language comprising routines
that are a preferred implementation of step 800. The
routines comprise the following, which are found in
section V, "symbols recognition": make_corr; correl_hem;
obj_funct; together with various definitions used by the
routines.

STEP 810: Create a list of probable symbols
sorted by likelihood. Based on the measure or measures
of comparison generated in step 800, a single list of
probable symbols sorted by likelihood is generated.

STEP 820: Choose the correct symbols and the
correct word based on the list, the database of previous
confusions and a dictionary. The symbols with greatest
likelihood are the candidates from which the correct
symbol is chosen.

The database of previous confusions provides
information that allows the correction of the choice of
the correct symbol based on previous incorrect identifi-
cations. The database of previous confusions comprises,
for each symbol, a list of other symbols which have been

SUBSTITUTE SHEET (RULE 26)

21

confused with the first symbol; for example, that the symbol "f" has often been confused with the symbol "b". When such an entry is found comprising previous confusions for a symbol in the list, the symbol or symbols that have previously been confused with the symbol in the list are added to the list. In accordance with the previous example, if the symbol "f" is found in the list, then the symbol "b" is added to the list.

An indication of the end of each word has been passed as output since step 715, described above. Based on the indication, the most likely word, comprising the most likely identifications for each symbol in the list, is identified.

The most likely word is checked against the dictionary. Preferably, the dictionary comprises both a general dictionary used for all users of the system and a personal dictionary for each user of the system. If an entry exists in the dictionary for the most likely word, the word is chosen as the correct identification.

If the most likely word is not found in the dictionary, all possible word combinations in the list are formed and each is checked against the dictionary. Among all such words which are found in the dictionary, the word with the highest likelihood is then chosen as the correct identification.

If none of the words is found in the dictionary, the most likely word is chosen as the correct identification.

STEP 830: Check to see if a correction has been entered. During the process of recognition, the user of the system is preferably provided with a visual indication of each symbol recognized.

After the end of a word is detected, the user of the system preferably is provided with a visual indication of the word recognized. The user may indicate manually that a given word was incorrectly recognized and

SUBSTITUTE SHEET (RULE 26)

22

may input a correction.

STEP 840: Update database of previous confu-
sions. Based on a manual correction entered in step 830
or an automatic correction based on the dictionary, the
database of previous confusions is updated. Based on a
manual correction, the personal dictionary is also updat-
ed if the corrected word is not found in the dictionary.

Preferred methods and apparatus for handwriting
recognition are described in the following applications,
the disclosure of which is hereby incorporated by refer-
ence: PCT/US92/08703; Israel 104575; PCT application
filed 31 January 1994 in the US Receiving Office by Ehud
Baron and Edward A. Wolfe.

It is appreciated that the particular embodi-
ment described in Appendix A is intended only to provide
an extremely detailed disclosure of the present invention
and is not intended to be limiting.

It is appreciated that various features of the
invention which are, for clarity, described in the con-
texts of separate embodiments may also be provided in
combination in a single embodiment. Conversely, various
features of the invention which are, for brevity, de-
scribed in the context of a single embodiment may also be
provided separately or in any suitable subcombination.

It will be appreciated by persons skilled in
the art that the present invention is not limited to what
has been particularly shown and described hereinabove.
Rather, the scope of the present invention is defined
only by the claims that follow:

SUBSTITUTE SHEET (RULE 26)

APPENDIX A

Recognition according a combination of signals.

Definitions and data structures

Board.H file

```
/* Function init_datatr ( portbase ) sets communication with data
   translation board via port portbase. It returns :
                   0 - communication was established ;
                   -1 - error on board (board is not exist). */
//int init_datatr ( int )  ;

int newcomp ( void ) ;

int read_ch ( int channel , int gain ) ;

//int read_point ( struct point * , int ) ;

/*int read_block ( struct point * , int max_numb_point ,
                int timeout_for_begin , int timeout_for_end ,
                int key_mouse_stop ) ; */

//int read_symbol ( struct point * , int , int ) ;

int mshit ( void ) ;

void close_datatr ( void ) ;

#define PORT_BASE    0x210

#define KEY_STOP       0x1
#define MOUSE_STOP     0x2
#define KEY_MOUSE_STOP 0x3

#define PEN_WAIT       0x1
#define PEN_NOWAIT     0x0


Data.H file
struct point_pen
{
   unsigned ax ;
   unsigned ay ;
   unsigned az ;
   unsigned pn ;
} ;
struct point_tablet
{
   int x ;
   int y ;
   int p ;
} ;
#define SYNCROBIT    0x80
```

SUBSTITUTE SHEET (RULE 26)

```
Datar.H file
#define PORT_BASE    0x210
#define CSR          0x0
#define GAIN         0x1
#define DAC0_LOW      0x2
#define DAC0_HIGH     0x3
#define DAC1_LOW      0x4
#define DAC1_HIGH     0x5

#define CHANNEL_AX    0x4
#define CHANNEL_AY    0x5
#define CHANNEL_AZ    0x6
#define CHANNEL_PN    0x7
#define STATUS        0xe
#define CHANNEL_EMPTY 0x0

#define IDREGISTER    0xf

#define GAIN_1       0x00
#define GAIN_2       0x40
#define GAIN_4       0x80
#define GAIN_8       0xc0

#define IER          0x21
#define IIR          0x20

#define IRQ0         0x08
#define IRQ1         0x09
#define IRQ2         0x0a
#define IRQ3         0x0b
#define IRQ4         0x0c
#define IRQ5         0x0d
#define IRQ6         0x0e
#define IRQ7         0x0f

/*struct point {
   unsigned ax ;
   unsigned ay ;
   unsigned az ;
   unsigned pn ;
} ;*/
#define MINUS_PEN    1700
/*#define PEN_UP      0x2
#define PEN_DOWN     0x4
#define PEN_THRSLD   200
#define EMPTY                0    */

#define BUFSIZE      0x80


#define TIME_COUNT   3000
#include <dos.h>
```

SUBSTITUTE SHEET (RULE 26)

Ser.H file
```
/*-------------------------------------------------------------*
     FILENAME:              SERIAL.H

             Some definitions used by SER.C


   *-------------------------------------------------------------*/


#define COM1            1
#define COM2            2
#define COM1BASE        0x3F8   /*  Base port address for COM1  */
#define COM2BASE        0x2F8   /*  Base port address for COM2  */


/*
     The 8250 UART has 10 registers accessible through 7 port addresses.
     Here are their addresses relative to COM1BASE and COM2BASE. Note
     that the baud rate registers, (DLL) and (DLH) are active only when
     the Divisor-Latch Access-Bit (DLAB) is on. The (DLAB) is bit 7 of
     the (LCR).

         o TXR Output data to the serial port.
         o RXR Input data from the serial port.
         o LCR Initialize the serial port.
         o IER Controls interrupt generation.
         o IIR Identifies interrupts.
         o MCR Send contorl signals to the modem.
         o LSR Monitor the status of the serial port.
         o MSR Receive status of the modem.
         o DLL Low byte of baud rate divisor.
         o DHH High byte of baud rate divisor.
*/
#define TXR        0      /*  Transmit register (WRITE) */
#define RXR        0      /*  Receive register  (READ)  */
#define IER        1      /*  Interrupt Enable        */
#define IIR        2      /*  Interrupt ID            */
#define LCR        3      /*  Line control            */
#define MCR        4      /*  Modem control           */
#define LSR        5      /*  Line Status             */
#define MSR        6      /*  Modem Status            */
#define DLL        0      /*  Divisor Latch Low       */
#define DLH        1      /*  Divisor latch High      */



/*-------------------------------------------------------------*
     Bit values held in the Line Control Register (LCR).
             bit            meaning
             ---            -------
             0-1            00=5 bits, 01=6 bits, 10=7 bits, 11=8 bits.
             2              Stop bits.
             3              0=parity off, 1=parity on.
             4              0=parity odd, 1=parity even.
             5              Sticky parity.
             6              Set break.
             7              Toggle port addresses.
   *-------------------------------------------------------------*/
#define NO_PARITY       0x00
#define EVEN_PARITY     0x18
#define ODD_PARITY      0x08
```

SUBSTITUTE SHEET (RULE 26)

11/9/2007, EAST Version: 2.1.0.14

```
/*-----------------------------------------------------------*
    Bit values held in the Line Status Register (LSR).
            bit                     meaning
            ---                     -------
            0                       Data ready.
            1                       Overrun error - Data register overwritten.
            2                       Parity error - bad transmission.
            3                       Framing error - No stop bit was found.
            4                       Break detect - End to transmission requested.
            5                       Transmitter holding register is empty.
            6                       Transmitter shift register is empty.
            7               Time out - off line.
 *-----------------------------------------------------------*/
#define RCVRDY          0x01
#define OVRERR          0x02
#define PRTYERR         0x04
#define FRMERR          0x08
#define BRKERR          0x10
#define XMTRDY          0x20
#define XMTRSR          0x40
#define TIMEOUT                 0x80


/*-----------------------------------------------------------*
    Bit values held in the Modem Output Control Register (MCR).
            bit         meaning
            ---         -------
            0                       Data Terminal Ready. Computer ready to go.
            1                       Request To Send. Computer wants to send data.
            2                       auxillary output #1.
            3                       auxillary output #2.(Note: This bit must be
                                    set to allow the communications card to send
                                    interrupts to the system)
            4                       UART ouput looped back as input.
            5-7                     not used.
 *-----------------------------------------------------------*/
#define DTR             0x01
#define RTS             0x02
#define MC_INT                  0x08


/*-----------------------------------------------------------*
    Bit values held in the Modem Input Status Register (MSR).
            bit                     meaning
            ---                     -------
            0                       delta Clear To Send.
            1                       delta Data Set Ready.
            2                       delta Ring Indicator.
            3                       delta Data Carrier Detect.
            4                       Clear To Send.
            5                       Data Set Ready.
            6                       Ring Indicator.
            7                       Data Carrier Detect.
 *-----------------------------------------------------------*/
#define CTS             0x10
#define DSR             0x20
```

SUBSTITUTE SHEET (RULE 26)

11/9/2007, EAST Version: 2.1.0.14

```
/*------------------------------------------------------------*
    Bit values held in the Interrupt Enable Register (IER).
          bit                    meaning
          ---                    -------
          0                      Interrupt when data received.
          1                      Interrupt when transmitter holding reg. empty.
          2                      Interrupt when data reception error.
          3                      Interrupt when change in modem status register.
          4-7                    Not used.
 *------------------------------------------------------------*/
#define RX_INT          0x01



/*------------------------------------------------------------*
    Bit values held in the Interrupt Identification Register (IIR).
          bit                    meaning
          ---                    -------
          0                      Interrupt pending
          1-2                    Interrupt ID code
                                 00=Change in modem status register,
                                 01=Transmitter holding register empty,
                                 10=Data received,
                                 11=reception error, or break encountered.
          3-7                    Not used.
 *------------------------------------------------------------*/
#define RX_ID           0x04
#define RX_MASK          0x07



/*
    These are the port addresses of the 8259 Programmable Interrupt
    Controller (PIC).
*/
#define IMR             0x21   /* Interrupt Mask Register port */
#define ICR             0x20   /* Interrupt Control Port       */



/*
    An end of interrupt needs to be sent to the Control Port of
    the 8259 when a hardware interrupt ends.
*/
#define EOI             0x20   /* End Of Interrupt */



/*
    The (IMR) tells the (PIC) to service an interrupt only if it
    is not masked (FALSE).
*/
#define IRQ3            0xF7  /* COM2 */
#define IRQ4            0xEF  /* COM1 */



/*
    The (IMR) tells the (PIC) to service an interrupt only if it
    is not masked (FALSE).
*/
#define IRQ3            0xF7  /* COM2 */
#define IRQ4            0xEF  /* COM1 */
```

SUBSTITUTE SHEET (RULE 26)

```
int flag;
int  SetSerial();
int  SetOthers(int Parity, int Bits, int StopBit);
int  SetSpeed(int Speed);
int  SetPort(int Port);
void init_serial(void);
void comm_off(void);

void setallport(int Port, int Speed, int Parity, int Bits, int StopBit);
int  putchport (char);
void putstrport(char *);
int  getchport(void);
void offport();
```

Serconst.H file

```
/*------------------------------------------------------------------*
    FILENAME:              SERCONST.H

            Some definitions used by SER.C


*------------------------------------------------------------------*/


/*
    The 8250 UART has 10 registers accessible through 7 port addresses.
    Here are their addresses relative to COM1BASE and COM2BASE. Note
    that the baud rate registers, (DLL) and (DLH) are active only when
    the Divisor-Latch Access-Bit (DLAB) is on. The (DLAB) is bit 7 of
    the (LCR).

        o TXR Output data to the serial port.
        o RXR Input data from the serial port.
        o LCR Initialize the serial port.
        o IER Controls interrupt generation.
        o IIR Identifies interrupts.
        o MCR Send contorl signals to the modem.
        o LSR Monitor the status of the serial port.
        o MSR Receive status of the modem.
        o DLL Low byte of baud rate divisor.
        o DHH High byte of baud rate divisor.
*/
#define TXR      0       /* Transmit register (WRITE) */
#define RXR      0       /* Receive register  (READ) */
#define IER      1       /* Interrupt Enable       */
#define IIR      2       /* Interrupt ID           */
#define LCR      3       /* Line control           */
#define MCR      4       /* Modem control          */
#define LSR      5       /* Line Status            */
#define MSR      6       /* Modem Status           */
#define DLL      0       /* Divisor Latch Low      */
#define DLH      1       /* Divisor latch High     */

#define DLAB     0x80    /* */
/*------------------------------------------------------------------*
    Bit values held in the Line Control Register (LCR).
        bit              meaning
        ---              -------
```

SUBSTITUTE SHEET (RULE 26)

11/9/2007, EAST Version: 2.1.0.14

0-1	00=5 bits, 01=6 bits, 10=7 bits, 11=8 bits.
2	Stop bits.
3	0=parity off, 1=parity on.
4	0=parity odd, 1=parity even.
5	Sticky parity.
6	Set break.
7	Toggle port addresses.

```
*-----------------------------------------------------------*/
#define NO_PARITY      0x00
#define EVEN_PARITY    0x18
#define ODD_PARITY     0x08



/*----------------------------------------------------------*
   Bit values held in the Line Status Register (LSR).
       bit              meaning
       ---              -------
       0                Data ready.
       1                Overrun error - Data register overwritten.
       2                Parity error - bad transmission.
       3                Framing error - No stop bit was found.
       4                Break detect - End to transmission requested.
       5·               Transmitter holding register is empty.
       6                Transmitter shift register is empty.
       7            Time out - off line.
    *-----------------------------------------------------------*/
#define RCVRDY      0x01
#define OVRERR      0x02
#define PRTYERR      0x04
#define FRMERR      0x08
#define BRKERR      0x10
#define XMTRDY      0x20
#define XMTRSR      0x40
#define TIMEOUT            0x80


/*----------------------------------------------------------*·
   Bit values held in the Modem Output Control Register (MCR).
       bit        meaning
       ---        -------
       0                Data Terminal Ready. Computer ready to go.
       1                Request To Send. Computer wants to send data.
       2                auxillary output #1.
       3                auxillary output #2.(Note: This bit must be
                        set to allow the communications card to send
                        interrupts to the system)
       4                UART ouput looped back as input.
       5-7              not used.
    *-----------------------------------------------------------*/
#define DTR        0x01
#define RTS        0x02
#define MC_INT            0x08


/*----------------------------------------------------------*
   Bit values held in the Modem Input Status Register (MSR).
       bit              meaning
       ---              -------
```

SUBSTITUTE SHEET (RULE 26)

```
        0                delta Clear To Send.
        1                delta Data Set Ready.
        2                delta Ring Indicator.
        3                delta Data Carrier Detect.
        4                Clear To Send.
        5                Data Set Ready.
        6                Ring Indicator.
        7                Data Carrier Detect.
*------------------------------------------------------------*/
#define CTS          0x10
#define DSR          0x20


/*----------------------------------------------------------*
Bit values held in the Interrupt Enable Register (IER).
        bit              meaning
        ---              -------
        0                Interrupt when data received.
        1                Interrupt when transmitter holding reg. empty.
        2                Interrupt when data reception error.
        3                Interrupt when change in modem status register.
        4-7              Not used.
*------------------------------------------------------------*/
#define RX_INT       0x01


/*----------------------------------------------------------*
Bit values held in the Interrupt Identification Register (IIR).
        bit              meaning
        ---              -------
        0                Interrupt pending
        1-2              Interrupt ID code
                         00=Change in modem status register,
                         01=Transmitter holding register empty,
                         10=Data received,
                         11=reception error, or break encountered.
        3-7              Not used.
*------------------------------------------------------------*/
#define RX_ID        0x04
#define RX_MASK      0x07


/*
    These are the port addresses of the 8259 Programmable Interrupt
    Controller (PIC).
*/
#define IMR          0x21  /* Interrupt Mask Register port */
#define ICR          0x20  /* Interrupt Control Port      */


/*
    An end of interrupt needs to be sent to the Control Port of
    the 8259 when a hardware interrupt ends.
*/
#define EOI          0x20  /* End Of Interrupt */


/*
```

SUBSTITUTE SHEET (RULE 26)

```
        The (IMR) tells the (PIC) to service an interrupt only if it
        is not masked (FALSE).
*/
/*unsigned char IRQ[8] = { ~0x01 , ~0x02 , ~0x04 , ~0x80 ,
                          ~0x10 , ~0x */
#define IRQ3        0xF7  /* COM2 */
#define IRQ4        0xEF  /* COM1 */

int SerSetPortBase ( int , unsigned * ) ;
int SerSetSpeed ( unsigned , long ) ;
int SerSetBitsParityStopBit ( unsigned , int , int , int ) ;
int SerPutChar ( unsigned , unsigned char ) ;
int SerPutString ( unsigned , unsigned char * ) ;
int SerInitBuffer ( unsigned ) ;
int SerGetChar ( unsigned ) ;
int SerTestDSR ( unsigned ) ;
int SerTestCTS ( unsigned ) ;

/*   int flag;
    int  SetSerial();
    int  SetOthers(int Parity, int Bits, int StopBit);
    int  SetSpeed(int Speed);
    int  SetPort(int Port);
    void init_serial(void);
    void comm_off(void);

    void setallport(int Port, int Speed, int Parity, int Bits, int StopBit);
    int  putchport (char);
    void putstrport(char *);
    int  getchport(void);
    void offport();
*/


Tablet.H file
#define PEN_DOWN                    1
#define PEN_UP                           0
#define PEN_OUTPROX                     99
#define TBL_WACOM_II            3
#define TBL_DATA_ASCII          1
#define TBL_DATA_BINARY             0
#define TBL_MODE_STREAM             3
#define TBL_MODE_SWITCH_STREAM              2
#define TBL_MODE_SUPRESSED          0
#define TBL_MODE_POINT              0
#define TBL_TYPE_ABSOLUTE       0
#define TBL_TYPE_RELATIVE           1
#define TBL_MILLIMETERS         0
#define TBL_INCHES              1
#define TBL_ALWAYS_TRANSMIT_YES     1
#define TBL_ALWAYS_TRANSMIT_NO      0
#define TBL_BAUD_19200              7
#define TBL_BAUD_9600               6
#define TBL_BAUD_4800               5
#define TBL_BAUD_2400               4
#define TBL_BAUD_1200               3
#define TBL_BAUD_600                2
#define TBL_BAUD_300                1
```

SUBSTITUTE SHEET (RULE 26)

```
#define TBL_BAUD_150                    0
#define TBL_PARITY_NONE                 0
#define TBL_PARITY_ODD                  1
#define TBL_PARITY_EVEN                 2
#define TBL_STOPBITS_1                  0
#define TBL_STOPBITS_2                  1
#define TBL_DSR_MONITOR_OFF             0
#define TBL_DSR_MONITOR_ON              1
#define TBL_DATALENGTH_7        0
#define TBL_DATALENGTH_8        1
#define TBL_TRANSFER_RATE_MAX           7
#define TBL_TRANSFER_RATE_100         6
#define TBL_TRANSFER_RATE_67          5
#define TBL_TRANSFER_RATE_50          4
#define TBL_TRANSFER_RATE_20          3
#define TBL_TRANSFER_RATE_10          2
#define TBL_TRANSFER_RATE_5           1
#define TBL_TRANSFER_RATE_1           0
#define TBL_ORIGINLOG_UPPER_LEFT        1
#define TBL_ORIGINLOG_LOWER_LEFT        0
#define TBL_DATA_TERMINATOR_CR_LF       2
#define TBL_DATA_TERMINATOR_LF          1
#define TBL_DATA_TERMINATOR_CR          0


int read_point_tablet_pen ( unsigned , int ,
                        struct point_tablet * , struct point_pen *[8] ) ;
int find_set_parameters_tablet ( int comport , unsigned *portbase ) ;
int init_tablet ( int port , unsigned *portbase , int command_set ,
            int data_format , int operation_mode , int origin_type ,
            int unit_mesure , int always_transmit , int speed ,
            int parity , int stopbit , int dsr_monitor ,
            int datalength , int transfer_rate , int orig_log ,
            int data_terminator , int max_x , int max_y ) ;
void close_tablet ( unsigned portbase ) ;
```

SUBSTITUTE SHEET (RULE 26)

I. Reading from device

```
/*      This procedure reads synchronized data from the graphic tablet and accelerometers   /*

int read_point_tablet_pen ( unsigned portbase , int read_pen ,
                            struct point_tablet *tablet ,
                            struct point_pen pen[8] )
{
    int ind_package = 0 , reply , debug[10] , i ;
    unsigned char package[7] = { 0 , 0 , 0 , 0 , 0 , 0 , 0 } ;
    if ( read_pen )
        read_point_pen ( &pen[0] ) ;
    i= 0 ;

/* Waiting for synchro-bit */

    do
    {
        if ( ( reply = SerGetChar ( portbase ) ) < 0 )
            return reply ;
        debug[i++] = reply ;
        if ( ( package[0] = (char) reply ) & SYNCROBIT )
            break ;
    } while ( ind_package++ < 10 ) ;

/* Error - No synchro-bit in 10 bytes */

    if ( ind_package >= 10 )
        return SER_SYNCROBIT ;

/* Read the next 6 bytes from tablet and 6 points from accelerometer */

    for ( ind_package = 1 ; ind_package < 7 ; ind_package++ )
    {
        if ( read_pen )
        {
            read_point_pen ( &pen[ind_package] ) ;
        }
        if ( ( reply = SerGetChar ( portbase ) ) < 0 )
            return reply ;
        package[ind_package] = (char) reply ;
    }

/* Read last point from accelerometer */

    if ( read_pen )
        read_point_pen ( &pen[ind_package] ) ;

/* Calculates the values of the signals for tablet */

    tablet->x = ( package[0] & 0x03 ) << 14 ;
    tablet->x += ( package[1] & 0x7f ) << 7 ;
    tablet->x += ( package[2] & 0x7f ) ;
    if ( package[0] & 0x04 )
        tablet->x = - tablet->x ;
    tablet->y = ( package[3] & 0x03 ) << 14 ;
    tablet->y += ( package[4] & 0x7f ) << 7 ;
    tablet->y += ( package[5] & 0x7f ) ;
```

SUBSTITUTE SHEET (RULE 26)

```
      tablet->p = 0 ;
      if ( ! ( package[0] & 0x40 ) )
        tablet->p = 99 ;
      if ( package[3] & 0x04 )
        tablet->y = - tablet->y ;
      if ( package[6] & 0x20 )
        tablet->p = ( package[6] & 0x1f ) ;
      return 0 ;
}
```

II. Pre-processing

```
/*      Two procedures: Normalization in time and filtering the input signals by smoothing
                                        */

void normal ( int num_old , float arr_old[] , int num_new , float arr_new[] )
{
  double koeff ;
  int ind_old , ind_new ;
  koeff = (double) ( num_old - 1 ) / (float) ( num_new - 1 ) ;
  arr_new[0] = arr_old[0] ;
  for ( ind_new = 1 ; ind_new < num_new - 1 ; ind_new ++ ) {
    ind_old = (int) ( floor ( koeff * ind_new ) ) ;
    arr_new[ind_new] = ( ind_old + 1 - koeff * ind_new ) * arr_old[ind_old] +
                ( koeff * ind_new - ind_old ) * arr_old[ind_old + 1] ;
    arr_new[ind_new] = arr_new[ind_new] ;
  }
  arr_new[ind_new] = arr_old[num_old-1] ;
}


float smooth1 ( int num , float z[] )
{
  int ind ;
  float temp ;
  float norma ;
  for (ind = 1 , norma = 0 ; ind < num - 1  ; ind++ ) {
    temp = ( z[ind -1]+z[ind]+z[ind+1] ) /3. ;
    norma += abbs ( z[ind] - temp ) ;
    z[ind] = temp ;
  }
  return norma ;
}
```

SUBSTITUTE SHEET (RULE 26)

III. Parameter's extraction

```
/*      Calculation of the parameters of a symbol from the input signals  */

int make_par ( char arg_ch )
{
  struct point {
    unsigned int x    : 12 ;
    unsigned int y    : 12 ;
    unsigned int z    : 12 ;
    unsigned int pen  :  4 ;
  } point , points[500];
  int read_next_symbol ( FILE * , struct point[] ) ;
  char file_name[40] ;
  int len , number_points = 0 ;
  FILE *in_file , *out_file[10] , *out_letter , *out_bin ;
  float  param[6][NUMBER_POINT] , sum_par[6][NUMBER_POINT] ;
  int index = 0 , max_point ;
  int ind , start ;
  int cur_x , cur_y , cur_z , cur_p ;
  float arr_x[MAX_POINT] , arr_y[MAX_POINT] , arr_z[MAX_POINT] , arr_p[MAX_POINT] ;

/* Initialization of the results arrays to zero  */

  for ( ind = 0 ; ind < 6 ; ind++ )
    for ( index = 0 ; index < NUMBER_POINT ; index++ ) {
      param[ind][index] = 0.0 ;
      sum_par[ind][index] = 0.0 ;
    }

/* Identification of the file of data   */

  sprintf ( file_name , "%03d.smb" , (int) arg_ch ) ;
  if ( ( in_file = fopen ( file_name , "rb" ) ) == NULL )
  {
    strcpy (ext_err,file_name);
    return -4 ;
  }
  start = 0 ;

/* Reading data from file   */

  while ( ( max_point = read_next_symbol ( in_file , points ) ) > 0 ) {
    for ( index = 0 ; index < max_point ; index++ ) {
      arr_x[index] = (float) points[index].x ;
      arr_y[index] = (float) points[index].y ;
      arr_z[index] = (float) points[index].z ;
      arr_p[index] = (float) points[index].pen ;
    }
    arr_p[0] = arr_p[max_point - 1] = 1 ;

    start++ ;
    number_points += max_point ;

/* Calling the procedure make_par_let for calculating parameters 1-6 */

    make_par_let ( arr_x , arr_y , arr_z , arr_p , param , max_point - 1 ) ;
```

SUBSTITUTE SHEET (RULE 26)

```
/* Calculating the average of each parameter */

   for ( ind = 0 ; ind < 6 ; ind++ )
     for ( index = 0 ; index < NUMBER_POINT ; index++ ) {
        sum_par[ind][index] += param[ind][index] ;
     }
  }
  for ( ind = 0 ; ind < 6 ; ind++ )
    for ( index = 0 ; index < NUMBER_POINT ; index++ )
      sum_par[ind][index] /= start ;

  sum_par[0][0] = (float) number_points / start ;
  fclose ( in_file ) ;

  /* write avg in Binary file */
  sprintf ( file_name , "%03d.par" , (int) arg_ch ) ;
  out_letter = fopen ( file_name , "wb+") ;
  for ( index = 0 ; index < 6 ; index++ )
    fwrite ( sum_par[index] , sizeof(float) , NUMBER_POINT , out_letter);

  fclose ( out_letter ) ;

  return  start ;


}


void make_par_let ( float arr_x[] , float arr_y[] , float arr_z[] ,
                  float arr_p[] , float param[6][NUMBER_POINT] , int max_point )
{
  float end_smooth;

  float new_arr_x[500] , new_arr_y[500] , new_arr_z[500] , new_arr_p[500] ;
  int ind , index ;

/* Call for pre-processing  */

  normal ( max_point , arr_x , NUMBER_POINT , new_arr_x ) ;
  normal ( max_point , arr_y , NUMBER_POINT , new_arr_y ) ;
  normal ( max_point , arr_z , NUMBER_POINT , new_arr_z ) ;
  normal ( max_point , arr_p , NUMBER_POINT , new_arr_p ) ;
  max_point = NUMBER_POINT ;
  for ( ind = 0 ; ind < max_point ; ind++ ) {
    arr_x[ind] = new_arr_x[ind] ;
    arr_y[ind] = new_arr_y[ind] ;
    arr_z[ind] = new_arr_z[ind] ;
    arr_p[ind] = new_arr_p[ind] ;
  }
  while ( ( end_smooth = smooth1 ( max_point , arr_x ) ) > NUMBER_POINT / 10 ) ;
  while ( ( end_smooth = smooth1 ( max_point , arr_y ) ) > NUMBER_POINT / 10 ) ;
  while ( ( end_smooth = smooth1 ( max_point , arr_z ) ) > NUMBER_POINT / 10 ) ;

/* Initialization of parameters */
  param[0][0] = (float) arr_p[0] ;
  param[1][0] = ( arr_z[0] - arr_z[0] ) ;
  param[2][0] = 0.0 ;
  param[3][0] = 0.0 ;
  param[4][0] = 0.0 ;
  param[5][0] = 0.0 ;
```

SUBSTITUTE SHEET (RULE 26)

```
param[0][1] = (float) arr_p[1] ;

/* Calculation of parameters */
  param[1][1] = ( arr_z[1] - arr_z[0] ) ;
  elev ( arr_x[2] - arr_x[0] , arr_y[2] - arr_y[0] , arr_z[2] - arr_z[0] ,
          &param[2][1] , &param[3][1] ) ;
  param[4][1] = 0.0 ;
  param[5][1] = 0.0 ;
  for ( index = 2 ; index < max_point - 2 ; index++ )  {
    param[0][index] = (float) arr_p[index] ;
    param[1][index] = ( arr_z[index] - arr_z[0] ) ;
    elev ( arr_x[index + 1] - arr_x[index - 1] , arr_y[index + 1] - arr_y[index - 1] , arr_z[index + 1] -
arr_z[index - 1] ,
          &param[2][index] , &param[3][index] ) ;
    angles ( arr_x[index + 2] - arr_x[index] ,
            arr_y[index + 2] - arr_y[index] ,
            arr_z[index + 2] - arr_z[index] ,
            arr_x[index] - arr_x[index - 2] ,
            arr_y[index] - arr_y[index - 2] ,
            arr_z[index] - arr_z[index - 2] ,
            &param[4][index] , &param[5][index] ) ;
    index = index ;
  }
  param[0][index] = (float) arr_p[index] ;
  param[1][index] = ( arr_z[index] - arr_z[0] ) ;
  elev ( arr_x[index + 1] - arr_x[index - 1] , arr_y[index + 1] - arr_y[index - 1] , arr_z[index + 1] -
arr_z[index - 1] ,
          &param[2][index] ; &param[3][index] ) ;
  param[4][index] = 0.0 ;
  param[5][index] = 0.0 ;
  index++ ;

/* Calculation of parameters for last point */
  param[0][index] = (float) arr_p[index] ;
  param[1][index] = ( arr_z[index] - arr_z[0] ) ;
  param[2][index] = 0.0 ;
  param[3][index] = 0.0 ;
  param[4][index] = 0.0 ;
  param[5][index] = 0.0 ;
}


/* Procedure elev calculates the SIN and COS of the angle of elevation */

void elev ( float x , float y , float z , float *cos_ug , float *sin_ug )
{
  float norma ;
  norma = (float) sqrt ( x * x + y * y + z * z ) ;
  if ( norma < .00001 ) {
    *cos_ug = 0.0 ;
    *sin_ug = 0.0 ;
    return ;
  }
  *cos_ug = ( (float) sqrt ( x * x + y * y ) ) / norma ;
  *sin_ug = z / norma ;
  return ;
}
```

SUBSTITUTE SHEET (RULE 26)

```c
/* Procedure angles calculates the SIN and COS of the angle β */

void angles ( float x1 , float y1 , float z1 , float x2 , float y2 , float z2 ,
              float *cos_ug , float *sin_ug )
{
  float norma1 , norma2 , x3 , y3 , z3 ;
  norma1 = ( float ) sqrt ( x1 * x1 + y1 * y1 + z1 * z1 ) ;
  norma2 = ( float ) sqrt ( x2 * x2 + y2 * y2 + z2 * z2 ) ;
  if ( norma1 < .0001 || norma2 < .0001 ) {
    *cos_ug = 0.0 ;
    *sin_ug = 0.0 ;
    return ;
  }
  *cos_ug = ( x1 * x2 + y1 * y2 + z1 * z2 ) / norma1 / norma2 ;
  x3 = ( y1 * z2 - z1 * y2 ) ;
  y3 = ( x2 * z1 - x1 * z2 ) ;
  z3 = ( x1 * y2 - x2 * y1 ) ;
  *sin_ug = ( (float) sqrt ( x3 * x3 + y3 * y3 + z3 * z3 ) ) / norma1 / norma2 ;
  return ;
}
```

IV. Training procedures

```c
/* Procedure for preliminary teaching */

int first_teach ( void )
{
  FILE *fp ;
  FILE *fpout;
  int i;
  char buf[4] , NdxStr[4] , symbols[256] ;
  int ndx = 0 , max_symb = 0 ;
  int num_sym;
  comment ("converting data files, please wait",0,1);

  if ( ( fp=fopen ( "symbols.dat" , "r" ) ) == NULL )
  {
     strcpy (ext_err,"symbols.dat");
     hide_comment ("converting data files, please wait",0);
     return (-4);
  }
  while ( fscanf ( fp , "%s" , buf ) > 0 )
    symbols[max_symb++] = buf[0] ;
  fclose ( fp ) ;

  fpout=fopen ("text.adp","w");

  for ( ndx = 0 ; ndx < max_symb ; ndx++ ) {
    sprintf ( NdxStr , "%03d" , ndx ) ;
    if ( ( num_sym=make_par ( symbols[ndx] ) ) <= 0 )
    {
       hide_comment ("converting data files, please wait",0);
       return (num_sym);
    }
    else for (i=0;i<num_sym;i++)
                fprintf (fpout,"%c",symbols[ndx]);
  }
  fclose (fpout);
  hide_comment ("converting data files, please wait",0);
  return (0);
}


/* procedure for adaptation of prototypes */

float huge *all_par[100] ;
int first_adap ( void )
{
  float old_rec , new_rec ;
  int count=0 , temp ;
  char *text ;
  char str[80];
  if ( ( temp = read_text ("try.txt", &text ) ) < 0 )
    return ( temp ) ;
  read_param ( ) ;
  new_rec = recogn ( "try.prl" , text , 0 , 0 ) ;
  sprintf (str,"%3f-before adaptation",new_rec);
  comment (str,-1,1);
  do {
```

SUBSTITUTE SHEET (RULE 26)

```
if (new_rec < 0 ) {
   hide_comment (str,-1);
   while ( all_par[temp] != NULL ) {
       farfree ( all_par[temp++] ) ;
   }
   return ((int) new_rec);
}
if ( new_rec  > .995 )
   break ;
old_rec = new_rec ;
new_rec = recogn ( "try.prl", text , 1 , 0 ) ;
if (new_rec <0 ) {
   hide_comment (str,-1);
   while ( all_par[temp] != NULL ) {
       farfree ( all_par[temp++] ) ;
   }
   return ((int) new_rec);
}
hide_comment (str,-1);
sprintf (str,"%3f- in adaptation",new_rec);
comment (str,-1,1);

new_rec = recogn ( "try.prl", text , 0 , 0 ) ;

hide_comment (str,-1);
sprintf (str,"%3f-after adaption",new_rec);
comment (str,-1,1);

if (new_rec < 0 ) {
    hide_comment (str,-1);
    while ( all_par[temp] != NULL ) {
      farfree ( all_par[temp++] ) ;
    }
    return ((int) new_rec);
}

} while ( fabs ( old_rec - new_rec ) > .005 & count++ < 9 ) ;
hide_comment (str,-1);
farfree ( text ) ;
while ( all_par[temp] != NULL ) {
  farfree ( all_par[temp++] ) ;
}
return 0 ;
}
```

V. Symbol's recognition

```
struct point {
  unsigned int x    : 12 ;
  unsigned int y    : 12 ;
  unsigned int z    : 12 ;
  unsigned int pen  :  4 ;
} ;

struct reply
{
  int   ndx;
  float weight;
};

float recogn ( char *file_pen , char *text , int adapt , int words )
{
  float old_rec , new_rec , probs[10][20] ;
  int count=0 ;
  char symbols[256] , buf[4] ;
  unsigned long ttt ;
  int max_symb;
  FILE *in_file , *file_symb , *temp_word ;
  int symb;
  unsigned long start_word , end_word ;
  float  param[6][NUMBER_POINT] ;
  int index = 0 , max_point ;
  struct reply *repl ;
  int temp;
  int Ngood=0;
  int ind , NumSymbols ,ndx;
  struct point symb_pnts [MAX_POINT];
  float arr_x[MAX_POINT] , arr_y[MAX_POINT] , arr_z[MAX_POINT] , arr_p[MAX_POINT] ;
  int map[256];
  int order=0;
  char letters[10][20],dict_wrds[10][20];
  int end_of_word=0;
  int wrdlen;
  float sum[10],maxsum,ndx_maxsum;
  char org_wrd[20],f_word[20];
  int txt_width;
  int i;
  if ( ( file_symb = fopen ( "symbols.dat" , "r" ) ) == NULL ) {
    strcpy (ext_err,"symbols.dat");
    return (-4);
  }

  for (ind=0;ind<256;ind++) map[ind]=-1;
  max_symb = 0 ;
  while ( fscanf ( file_symb , "%s" , buf ) > 0 )
  {
    map [buf[0]]=max_symb;
    symbols[max_symb++] = buf[0] ;
  }
  fclose ( file_symb ) ;
  symbols[max_symb] = 0 ;

  for ( ind = 0 ; ind < 6 ; ind++ )
```

SUBSTITUTE SHEET (RULE 26)

```
    for ( index = 0 ; index < NUMBER_POINT ; index++ ) {
      param[ind][index] = 0.0 ;
    }
  if ( ( in_file = fopen ( file_pen , "rb" ) ) == NULL )
  {
    strcpy (ext_err,file_pen);
    return -4 ;
  }
  index = 0 ;
  NumSymbols = 0 ;
  symb=-1;
          if (adapt)
            repl = make_corr ( param , symbols , symb) ;
          else {
            repl = make_corr ( param , symbols , -1) ;
          }
          if (repl[0].ndx<0)
            return ( repl[0].weight);

          if (repl[0].ndx==symb)
            Ngood++;
          else
            Ngood = Ngood ;
      }
  fclose ( in_file ) ;
  if (NumSymbols==0) return 0;
    else return (Ngood/(float)NumSymbols) ;
}


/* Calculation of the similarity of all the parameters of all the prototypes and the
   symbol to be recognized */

extern float huge *all_par[100] ;

struct reply
{
  int   ndx;
  float weight;
};

static int comm_count = 0 , abs_count = 0 ;
int obj_funct ( float [100][7] , int , int , float [100] , float [7] , int [10] ) ;
float correl_hem ( float [NUMBER_POINT] , float [NUMBER_POINT] , float ) ;
float correl ( float [NUMBER_POINT] , float [NUMBER_POINT] ) ;
struct reply *make_corr ( float cur_par[6][NUMBER_POINT] , char *symbols ,int symb)
{
  FILE *cur_file ;
  int ind_repl , ind_corrct , ind , max_symb , ind_symb , index ;
  struct reply arr_repl[30];
  int arr_ind[10];
  float res[100] , nres[7] , old_max_pnt = cur_par[0][0] , com_wight ;
  float old_max_pnt2 , corr[100][7] , tmp_par[6][NUMBER_POINT] ;
  char buf[8] ;
  int iterat;
  struct reply rt;
  int i,j;
```

```
max_symb = strlen ( symbols ) ;
for ( ind_symb = 0 ; ind_symb < max_symb ; ind_symb++ ) {
  for ( i = 0 ; i < 6 ; i++ )
    for ( j = 0 ; j < NUMBER_POINT ; j++ )
    tmp_par[i][j] = all_par[ind_symb][i*100+j] ;
  if ( tmp_par[0][0] > 0 ) {
    cur_par[0][0] = old_max_pnt ;
    corr[ind_symb][N_PAR-1] = 1.0 * ( 1 -
            min ( fabs ( tmp_par[0][0] - cur_par[0][0] ) / cur_par[0][0] , 1 ) ) ;

    old_max_pnt = cur_par[0][0] ;
    tmp_par[0][0] = 1. ;
    cur_par[0][0] = 1. ;
    corr[ind_symb][0] = correl_hem ( cur_par[0] , tmp_par[0] , .9 ) ;
    for ( ind = 1 ; ind < N_PAR - 1; ind++ ) {
        corr[ind_symb][ind] = correl ( cur_par[ind] , tmp_par[ind] ) ;
    }
  }
  else
    for ( ind = 1 ; ind < N_PAR - 1; ind++ ) {
        corr[ind_symb][ind] = 0.0 ;
    }
}

if (symb<0)
{
  index = obj_funct ( corr , max_symb , N_PAR , res , nres , arr_ind ) ;
  iterat=20;
}
else
{
  sprintf ( buf , "%03d.par" , (int) symbols[symb] ) ;
  for ( i = 0 ; i < 6 ; i++ )
      for ( j = 0 ; j < NUMBER_POINT ; j++ )
      tmp_par[i][j] = all_par[symb][i*100+j] ;
  iterat=0;
  while ( (index=obj_funct (corr,max_symb,N_PAR,res,nres, arr_ind))>0
        && (arr_ind[0]!=symb))
  {

      if (iterat>19) break;
      for (ind=0, ind_corrct=0; ind<N_PAR-1 ; ind++)
        if (corr[symb][ind]<0.95 * nres[ind])
        {
          ind_corrct++;
          for (index=0; index < NUMBER_POINT ; index++)
              tmp_par[ind][index] = tmp_par [ind][index]*.9
                                    +cur_par[ind][index]*.1;
        }
        if (corr[symb][ind]<0.95 * nres[ind])
        {
          ind_corrct++;
          tmp_par[0][0] = tmp_par[0][0] * .9 + old_max_pnt * .1 ;
        }
        if (!ind_corrct) {
          iterat = 20 ;
          break;
        }
```

SUBSTITUTE SHEET (RULE 26)

```
            iterat++;
            cur_par[0][0]= old_max_pnt;
            corr[symb][N_PAR-1]=1-fabs(tmp_par[0][0]-cur_par[0][0])/ cur_par[0][0];
            old_max_pnt = cur_par[0][0];
            old_max_pnt2= tmp_par[0][0];
            tmp_par[0][0] = 1.;
            cur_par[0][0] = 1.;
            corr[symb][0] = correl_hem ( cur_par[0] , tmp_par[0] , .9 ) ;
            for ( ind = 1 ; ind < N_PAR - 1 ; ind++ ) {
               corr[symb][ind] = correl ( cur_par[ind] , tmp_par[ind] ) ;
            }
            cur_par[0][0] = old_max_pnt ;
            tmp_par[0][0] = old_max_pnt2 ;
        } /* while */
    } /* else */

    if ((iterat<20) && (index>0) && (iterat>0))
    {
        cur_file = fopen ( buf , "w+b" ) ;
        for ( index = 0 ; index < N_PAR -1 ; index++ )
          fwrite ( tmp_par[index] , sizeof ( float ) , NUMBER_POINT , cur_file ) ;
        fclose ( cur_file ) ;
         for ( i = 0 ; i < 6 ; i++ )
             for ( j = 0 ; j < NUMBER_POINT ; j++ )
             all_par[symb][i*100+j] = tmp_par[i][j] ;
    }
    index = min ( index , 9 ) ;
    arr_ind[index]=-1;
    res[arr_ind[index]]=-1;
    for (i=0;i<=index;i++)
    {
        arr_repl[i].ndx=arr_ind[i];
        arr_repl[i].weight=-res[arr_ind[i]];
    }
    return arr_repl ;
}




/* Calculation of correlation between two vectors */

float correl ( float first[NUMBER_POINT] , float second[NUMBER_POINT] )
{
   float sumxy = 0.0 , sumx = 0.0 , sumy = 0.0 ,  sumx2 = 0.0 , sumy2 = 0.0 ;
   int i_d , i_s ;
   for ( i_s = 0 ; i_s < NUMBER_POINT ; i_s++ ) {
       sumxy += first[i_s] * second[i_s] ;
       sumx  += first[i_s] ;
       sumy  += second[i_s] ;
       sumx2 += first[i_s] * first[i_s] ;
       sumy2 += second[i_s] * second[i_s] ;
    }
    if ( ( sumx2 - sumx * sumx / NUMBER_POINT ) < 0 ||
         ( sumy2 - sumy * sumy / NUMBER_POINT ) < 0 )
      return 0 ;
    if ( ( sumxy = ( sumxy - sumx * sumy / NUMBER_POINT ) /
```

SUBSTITUTE SHEET (RULE 26)

```
        sqrt ( sumx2 - sumx * sumx / NUMBER_POINT ) /
        sqrt ( sumy2 - sumy * sumy / NUMBER_POINT ) ) < .5 )
    return 0 ;
  return sumxy ;
}



/* Similarity function for the parameter of pen up/down */

float correl_hem ( float par1[NUMBER_POINT] , float par2[NUMBER_POINT] , float border )
{
  int index ;
  float result = 0.0 ;
  for ( index = 1 ; index < NUMBER_POINT ; index++ )
    result += fabs ( par1[index] - par2[index] ) ;
  result /= NUMBER_POINT ;
  result = 1 - result ;
  if ( result < border )
    return 0 ;
  return result ;
}


/* Selection of the list of symbols that are likely to be the symbol to be recognized */

int obj_funct ( float arr[100][7] , int n_symb , int n_par ,
                float res[100] , float nres[7] , int arrindex[30] )
{
  int ind_s , ind_p , ind_arr = 0 ;
  float max_res = 0.0 , cur_res , abs_res = 0.0 ;
  int result = -1 ;
  for ( ind_s = 0 ; ind_s < n_symb ; ind_s++ ) {
    for ( ind_p = 0 , cur_res = 0.0 ; ind_p < n_par ; ind_p++ )
      cur_res += arr[ind_s][ind_p] ;
    res[ind_s] = cur_res ;
    if ( cur_res > max_res ) {
      result = ind_s ;
      max_res = cur_res ;
    }
  }
  abs_res = max_res * .85 ;
  do {
    arrindex[ind_arr++] = result ;
    res[result] = - res[result] ;
    for ( ind_s = 0 , max_res = 0.0 ; ind_s < n_symb ; ind_s++ )
        if ( res[ind_s] > max_res ) {
          result = ind_s ;
          max_res = res[ind_s] ;
        }
  } while ( max_res > abs_res && ind_arr < 30 ) ;
  for ( ind_p = 0 ; ind_p < n_par ; ind_p++ )
    for ( ind_s = 0 , nres[ind_p] = -5 ; ind_s < n_symb ; ind_s++ )
      nres[ind_p] = max ( arr[ind_s][ind_p] , nres[ind_p] ) ;
  return ind_arr ;
}
```

SUBSTITUTE SHEET (RULE 26)

46

C L A I M S

1.   Information input apparatus comprising:

    body supported apparatus for sensing voluntary body motions and providing an output indication thereof;

    a symbol output interpreter operative to utilize said output indication for providing symbol outputs; and

    a motion output interpreter operative to utilize said output indication for providing motion control outputs.

2.   Information input apparatus according to claim 1 and wherein said output indication represents features of body motion including features which are characteristic of the individual.

3.   Information input apparatus according to either of claims 1 and 2 and also comprising a mode selector operative to cause a selected one of the symbol output interpreter and the motion output interpreter to function.

4.   Information input apparatus according to any of the preceding claims and wherein said body supported apparatus is a hand held device.

5.   Information input apparatus according to any of the preceding claims and wherein said body supported apparatus is a generally pen-shaped device.

6.   Information input apparatus according to claim 5 and wherein said generally pen-shaped device is operative to provide a visible writing function.

SUBSTITUTE SHEET (RULE 26)

47

7.   Information input apparatus according to any of
the preceding claims and also comprising an object whose
motion is controlled by said motion control outputs.

8.   Information input apparatus according to claim
7 ˎand wherein said object is a graphic object displayed
on a display.

9.   Information input apparatus according to claim
7 and wherein said object is a physical object.

10.   Information input apparatus according to any of
the preceding claims and wherein said symbol outputs
represent alphanumeric symbols.

11.   Information input apparatus according to any of
the preceding claims and wherein said symbol outputs
represent a sensory quality.

12.   Information input apparatus according to claim
7 and also comprising a computer, having a location input
and a symbol input, and a display operated by said com-
puter and wherein said symbol outputs represent informa-
tion to be displayed on said display and said motion
outputs are supplied to said location input and are
employed by the computer to govern the location of said
information on said display.

13.   Information input apparatus according to claim
7 or claim 12 and wherein said symbol outputs include
function commands.

14.   A method by which a manipulable device provides
an output indication representing its own angular motion,
the method comprising:
        recording   actual   acceleration   data   from   a

SUBSTITUTE SHEET (RULE 26)

48

plurality of accelerometers mounted in the manipulable device;

generating predicted acceleration data on the basis of hypothetical angular motion information;

comparing the predicted acceleration data to the actual acceleration data;

computing improved hypothetical angular motion information;

while the predicted acceleration data differs significantly from the actual acceleration data, repeating the generating, comparing and computing steps; and

providing an output indication of the improved hypothetical angular motion information.

15.   A method according to claim 14 wherein the angular motion information includes angular displacement information, angular velocity information and angular acceleration information.

16.   A method according to claim 14 or claim 15. and also comprising computing linear motion information from the improved hypothetical angular motion information and from the actual acceleration data.

17.   A method according to any of claims 14 - 16 wherein recording comprises recording from at least four accelerometers mounted in the manipulable device, wherein the accelerometers each have a center of mass and wherein the centers of mass do not lie within a single plane.

18.   A method according to any of the preceding claims 14 - 17 and also comprising receiving the output indication of the improved hypothetical angular motion information and manipulating an object in accordance therewith.

SUBSTITUTE SHEET (RULE 26)

49

19.   An accelerometer array mounted in a manipulable
device and comprising:
        at least four accelerometers each having a
center of mass, wherein the centers of mass do not lie
within a single plane; and
        a manipulable device motion computer receiving
input from the accelerometers and generating an output
signal indicative of the motion of the manipulable de-
vice.

20.   Apparatus according to claim 19 wherein the
manipulable device motion computer is operative to per-
form the following steps:
        recording actual acceleration data from the
accelerometers;
        generating predicted acceleration data on the
basis of hypothetical angular motion information;
        comparing the predicted acceleration data to
the actual acceleration data;
        computing improved hypothetical angular motion
information;
        while the predicted acceleration data differs
significantly from the actual acceleration data, repeat-
ing the generating, comparing and computing steps; and
        providing an output indication of the improved
hypothetical angular motion information.

21.   Apparatus according to any of claims 19 – 20
and also comprising an object manipulator receiving the
output signal indicative of the motion of the manipulable
device and manipulating an object in accordance there-
with.

22.   An information input method comprising:
        sensing voluntary body motions and providing an
output indication thereof;

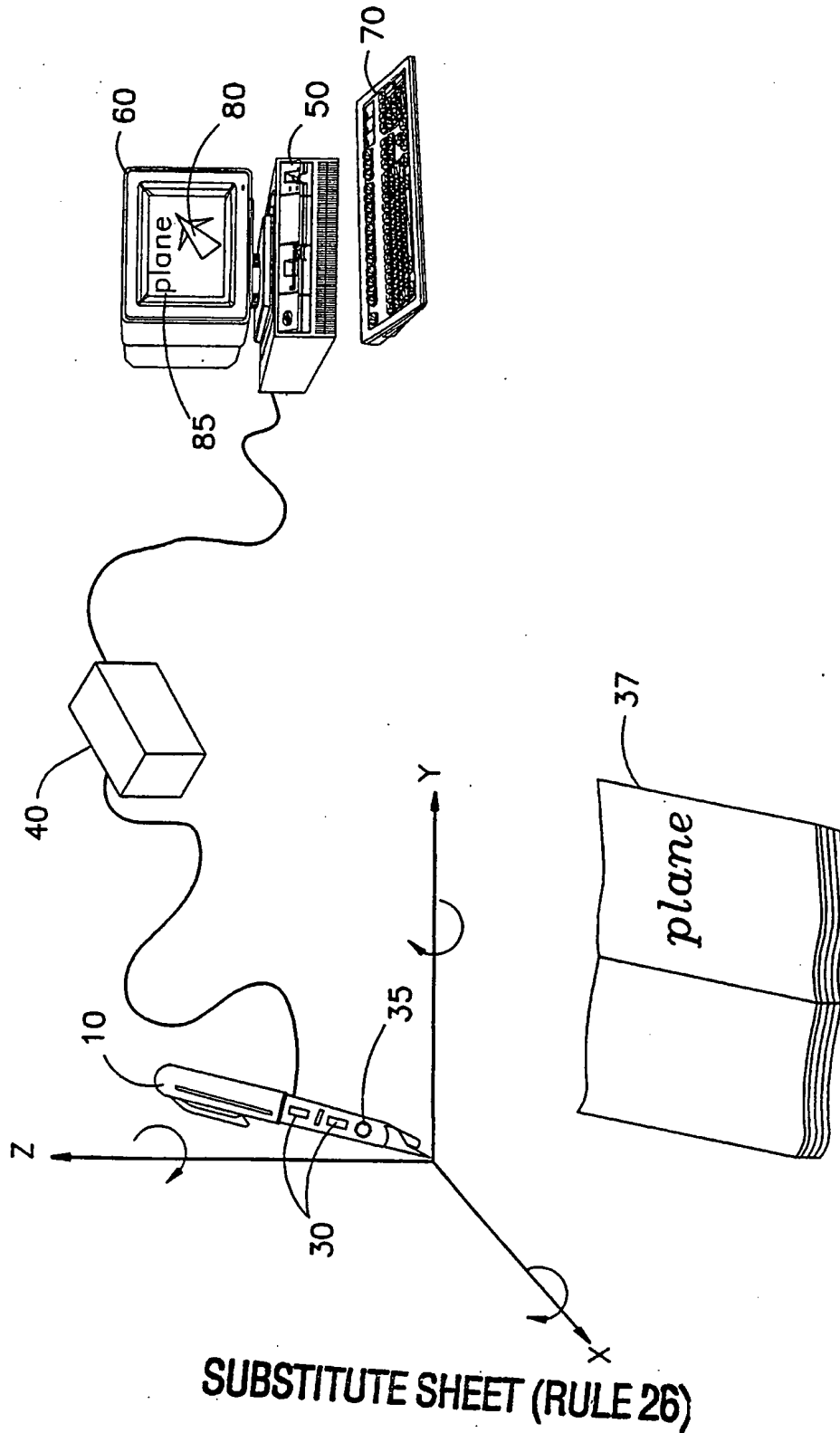# SUBSTITUTE SHEET (RULE 26)

50

　　　　utilizing said output indication for providing
symbol outputs; and

　　　　utilizing said output indication for providing
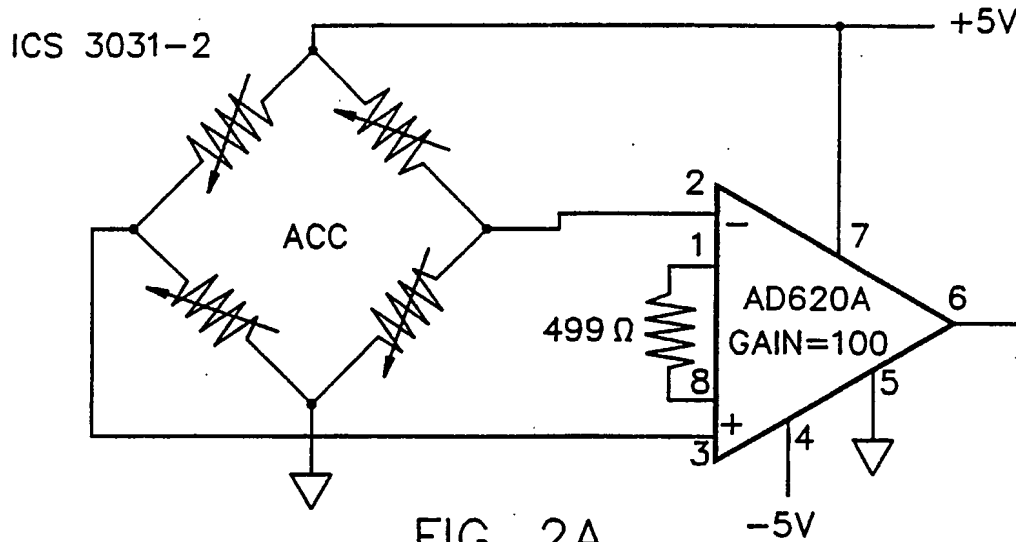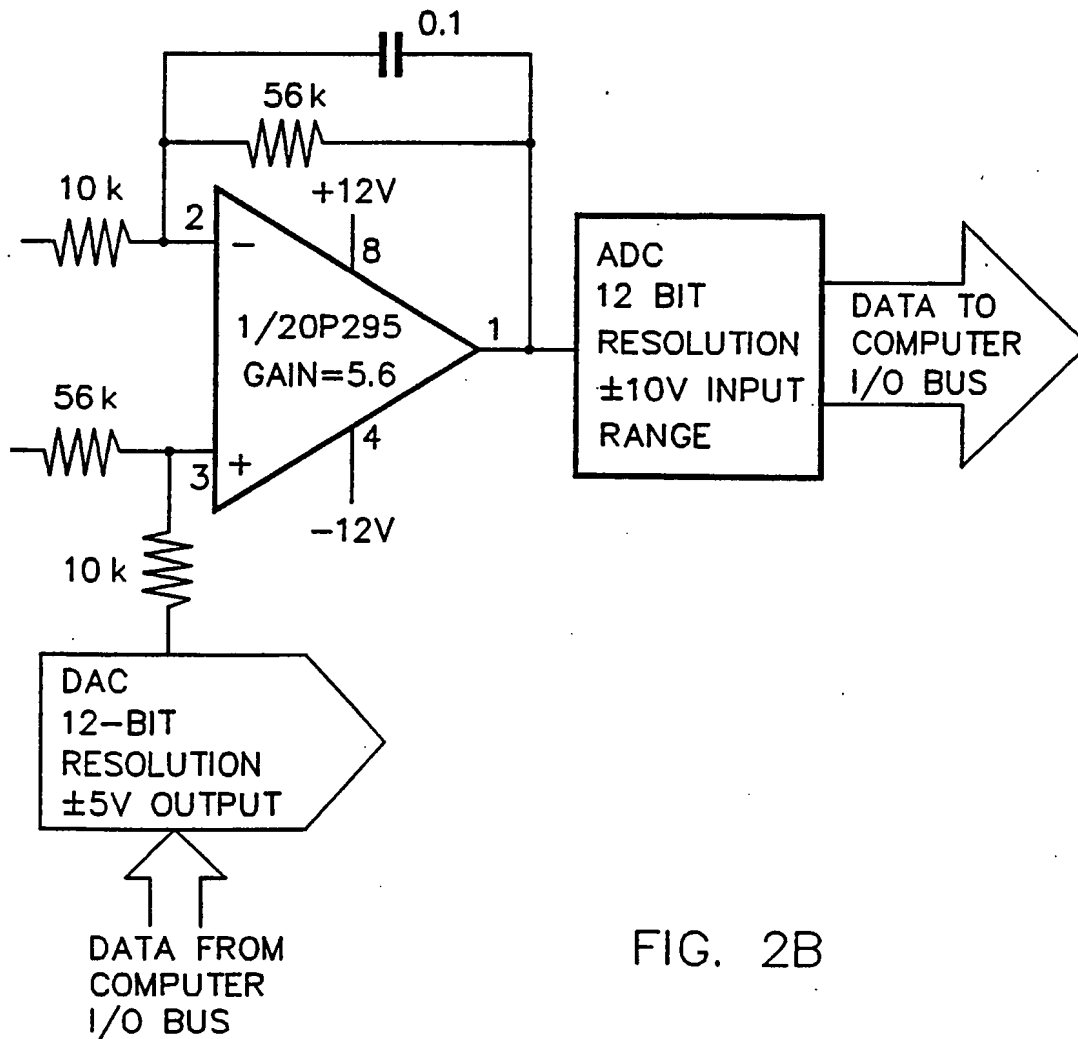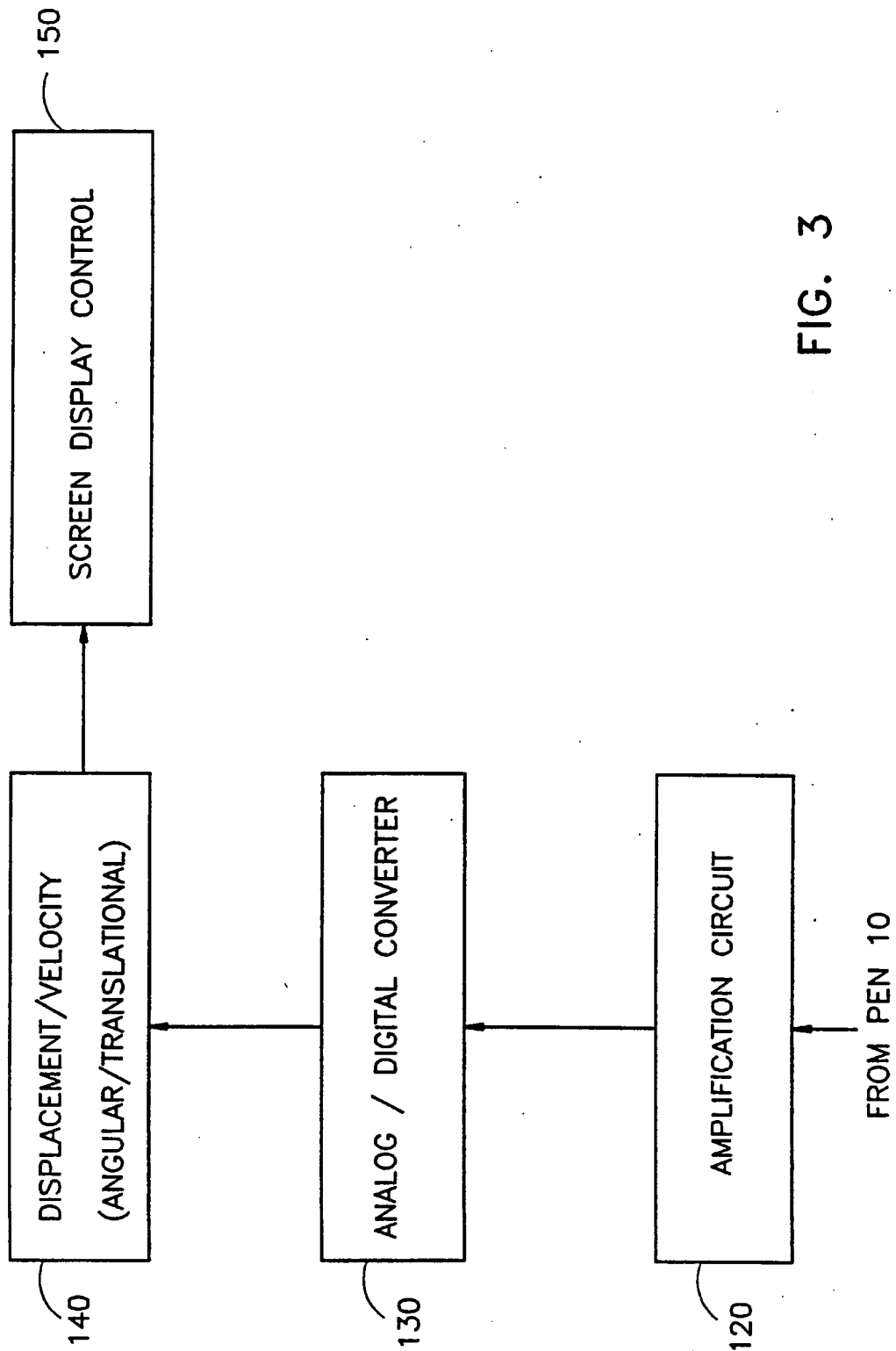motion control outputs.

SUBSTITUTE SHEET (RULE 26)

11/9/2007, EAST Version: 2.1.0.14

FIG. 1

ICS 3031-2

ACC

499 Ω

AD620A
GAIN=100

+5V

-5V

FIG. 2A

0.1

56 k

10 k

+12V

1/20P295
GAIN=5.6

-12V

56 k

10 k

ADC
12 BIT
RESOLUTION
±10V INPUT
RANGE

DATA TO
COMPUTER
I/O BUS

DAC
12-BIT
RESOLUTION
±5V OUTPUT

DATA FROM
COMPUTER
I/O BUS

FIG. 2B

FIG. 3

4/10

FIG. 4

11/9/2007, EAST Version: 2.1.0.14

5/10

```
                                              480
    ┌─────────────────────────────┐
    │  SET  INITIAL  PARAMETERS   │
    └─────────────────────────────┘
                  │
                  ▼
                 ◯ ◀────────────┐
                  │             │
                  ▼             │
    ┌─────────────────────────┐ │       480
    │ COMPUTE  DIFFERENTIAL   │ │
    │ ACCELERATION  FROM  MODEL│ │    FIG.  5A
    └─────────────────────────┘ │
                  │             │
                  ▼             │
              ╱───────╲    484  │
           ╱   DIFFERENCE  ╲    │
         ╱  BETWEEN  COMPUTED ╲ │    Y
        ╱ AND  CURRENT  LESS  THEN╲──────▶ REPORT  RESULT
         ╲  PREDETERMINED      ╱  │
           ╲   AMOUNT  ?    ╱     │
              ╲───────╱          │
                  │ N            │
                  ▼              │
                             486 │
    ┌───────────────────────────┐│
    │ COMPUTE  CHANGES  IN  ESTIMATED ││
    │ ANGLES  ACCORDING  TO  GRADIENT ││
    │         METHOD            ││
    └───────────────────────────┘│
                  │              │
                  ▼          488 │
    ┌───────────────────────────┐│
    │  COMPUTE  NEW  PARAMETERS  ├┘
    └───────────────────────────┘
```
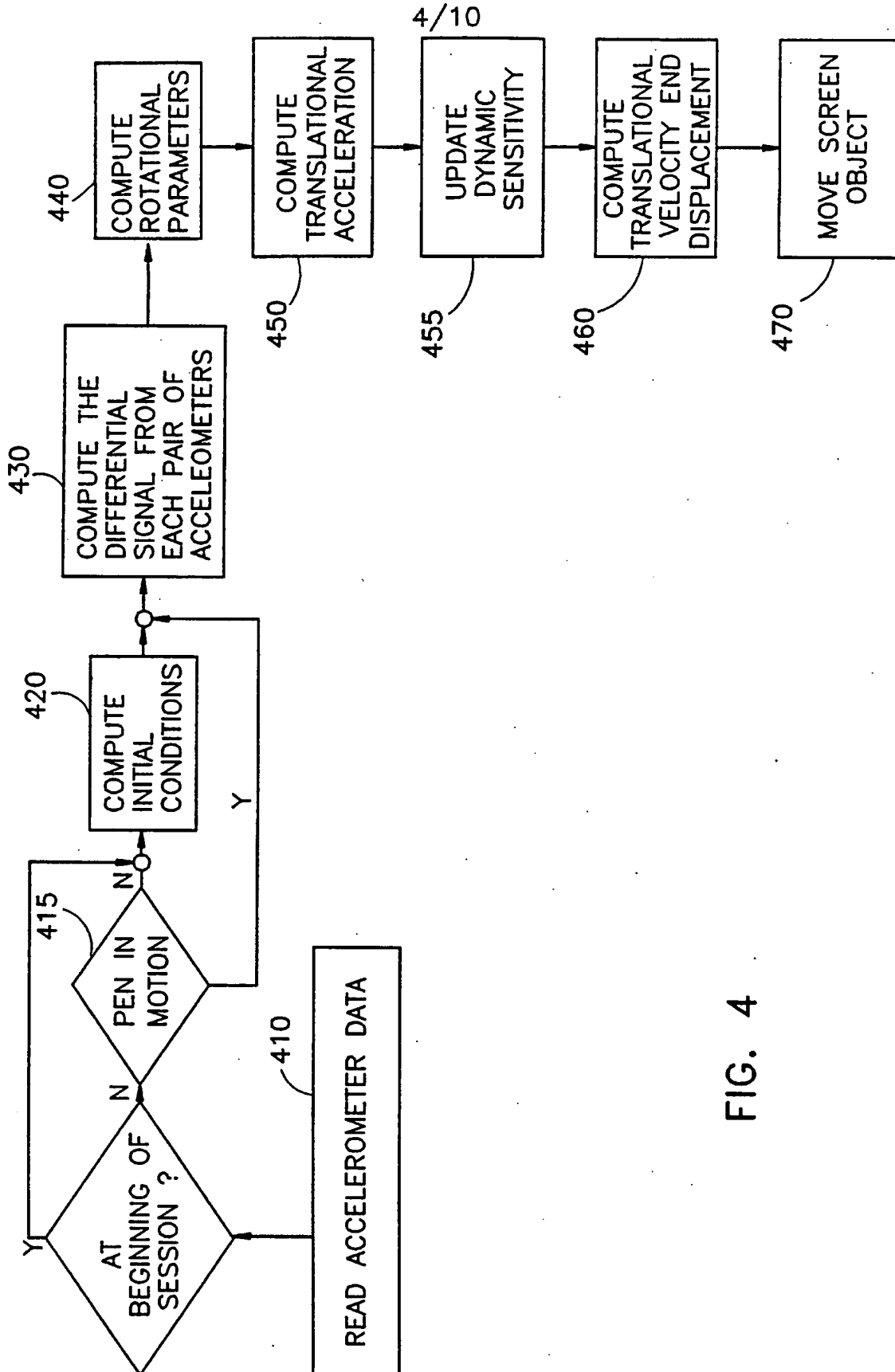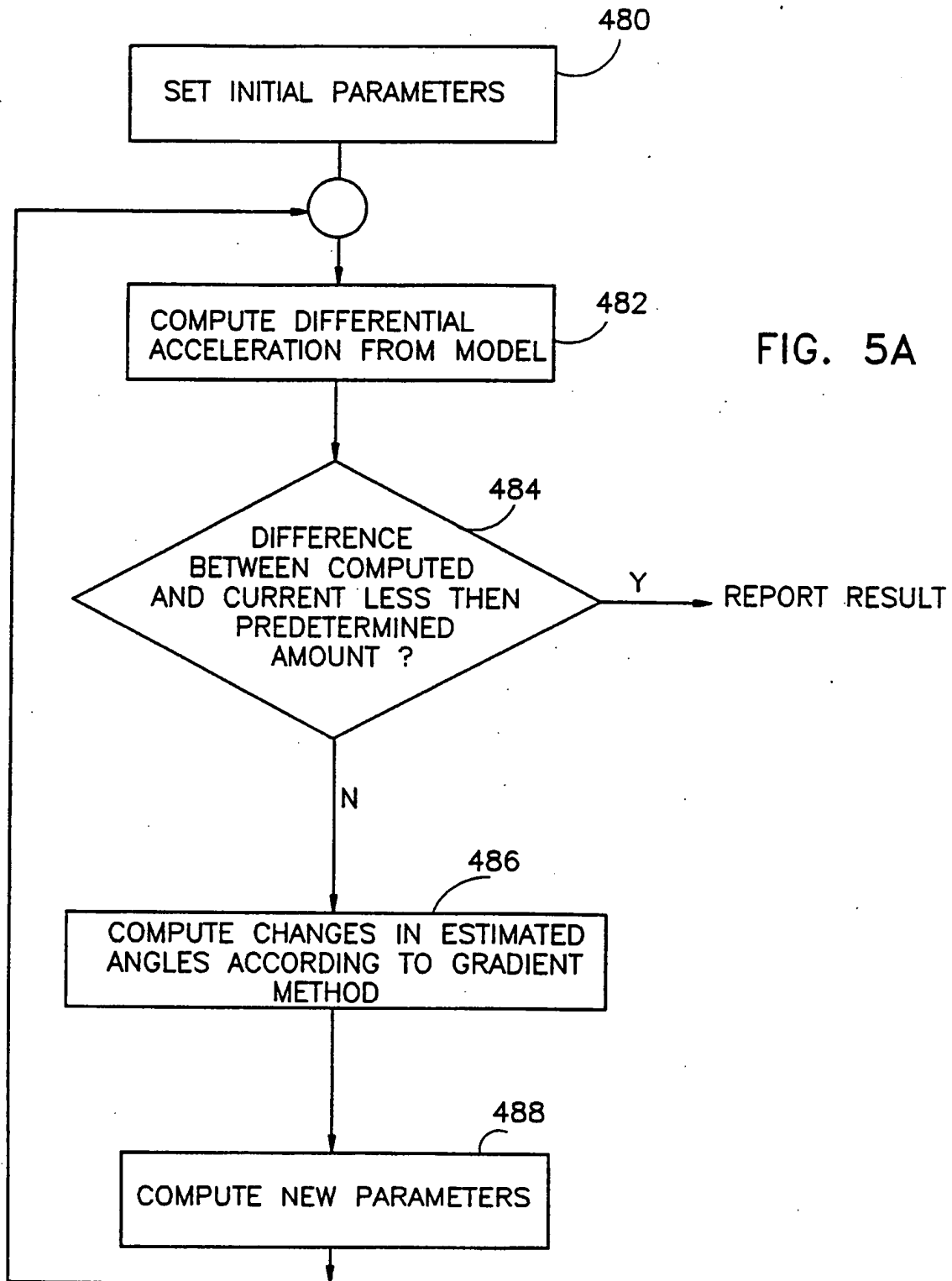
SUBSTITUTE SHEET (RULE 26)

EQUATION 490

$$\frac{d^2R}{dt^2} = [\frac{\partial A}{\partial \alpha} \cdot \frac{d^2\alpha}{dt^2} + \frac{\partial A}{\partial \beta} \cdot \frac{d^2\beta}{dt^2} \cdot \frac{\partial A}{\partial \gamma} + \frac{d^2\gamma}{dt^2} +$$

$$+ \frac{\partial^2 A}{\partial \alpha^2} \cdot (\frac{d\alpha}{dt})^2 + \frac{\partial^2 A}{\partial \beta^2} \cdot (\frac{d\beta}{dt})^2 + \frac{\partial^2 A}{\partial \gamma^2} \cdot (\frac{d\gamma}{dt})^2 +$$

$$+ 2 \frac{\partial^2 A}{\partial \alpha \partial \beta} \cdot \frac{d\alpha}{dt} \frac{d\beta}{dt} + 2 \frac{\partial^2 A}{\partial \alpha \partial \gamma} \cdot \frac{d\alpha}{dt} \frac{d\gamma}{dt} + 2 \frac{\partial^2 A}{\partial \beta \partial \gamma} \cdot \frac{d\beta}{dt} \frac{d\gamma}{dt}] * r$$

EQUATION 492

$$K(\varphi) = A \cdot K_0$$

EQUATION 494

$$U_{est} = K_0^T \cdot A^T \cdot \frac{d^2R}{dt^2}$$

EQUATION 496

$$\varphi(t) = \varphi_0' + \varphi_0 t + \frac{\varphi_0''}{2} t^2 + pt^3$$

$$\varphi'(t) = \varphi_0' + \varphi_0'' t + 3pt^2$$

$$\varphi''(t) = \varphi_0'' + 6pt$$
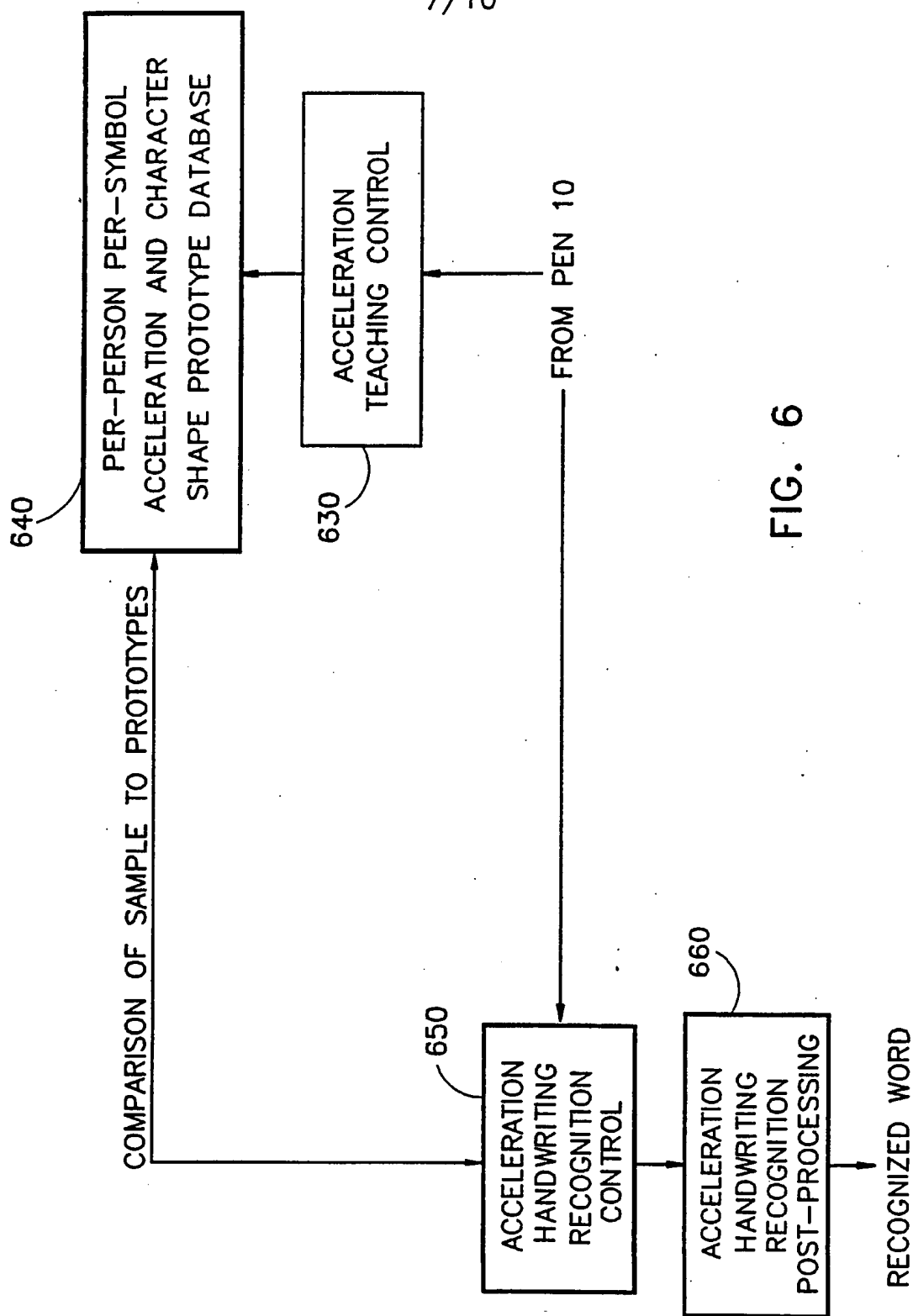
FIG. 5B

FIG. 6

11/9/2007, EAST Version: 2.1.0.14

FIG. 7A

710 READ ACCELEROMETER DATA

712 IDENTIFY PEN-SURFACE CONTACT TERMINATION

715 IDENTIFY INDIVIDUAL SYMBOLS & WORDS

720 NORMALIZE ACCELEROMETER DATA

730 FILTER ACCELEROMETER DATA

740 PARAMETRIZE ACCELEROMETER DATA

750 GENERALIZE PARAMETERS

760 UPDATE PER-PERSON PER-SYMBOL ACCELERATION PROTOTYPE DATABASE

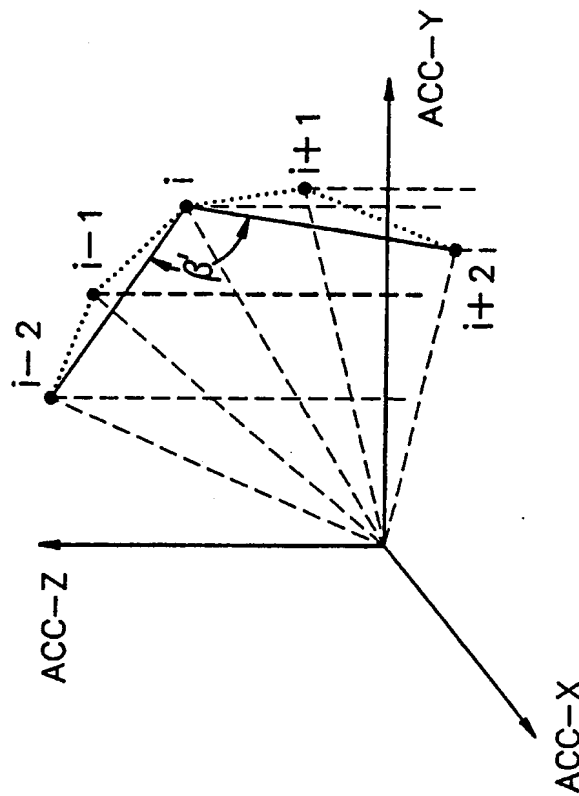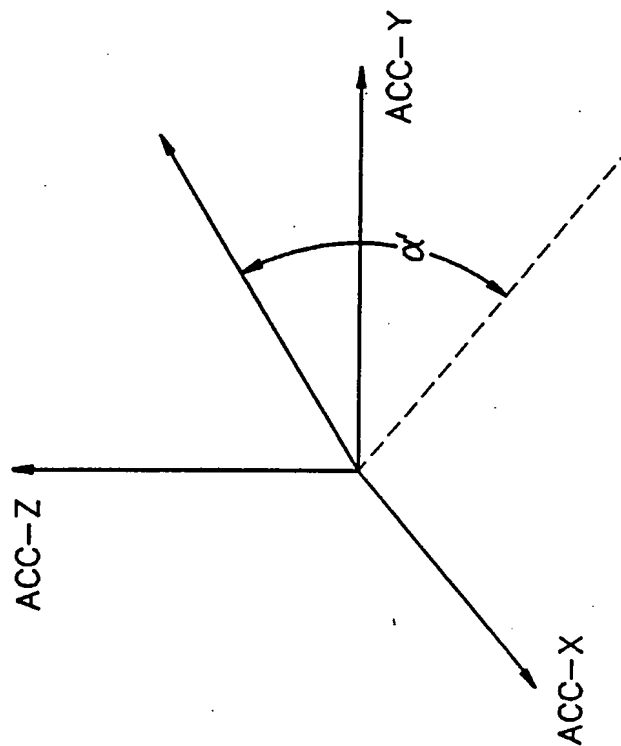SUBSTITUTE SHEET (RULE 26)

11/9/2007, EAST Version: 2.1.0.14
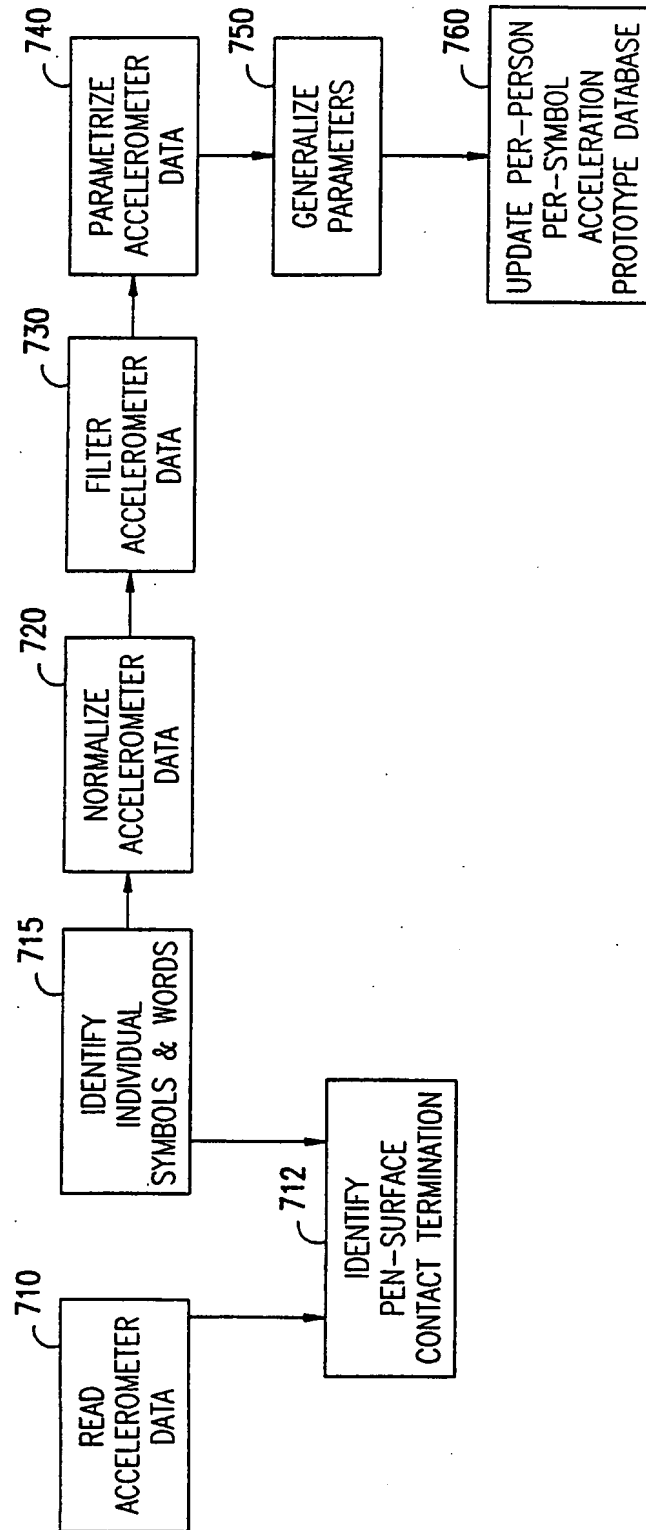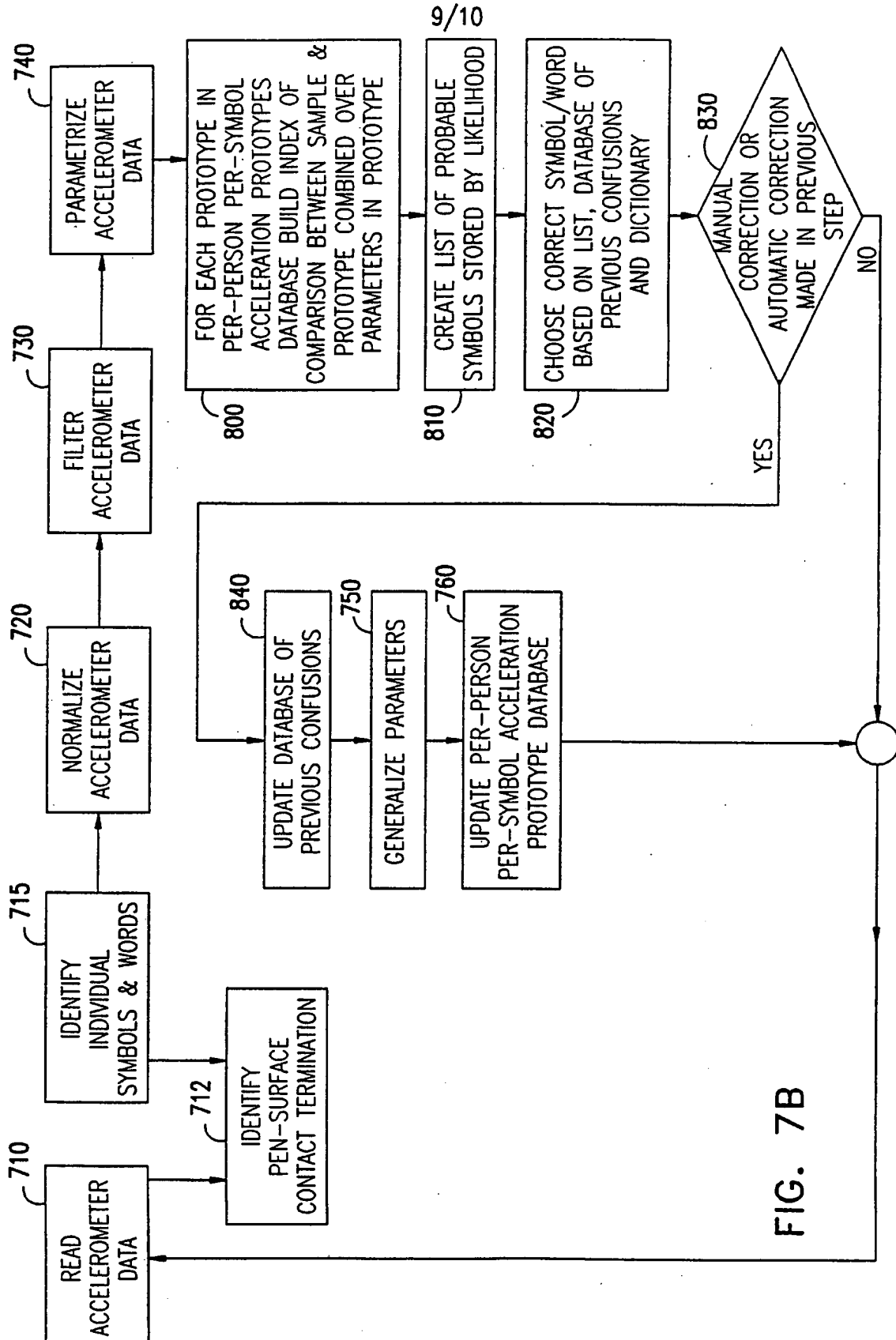
FIG. 7B

FIG. 8B

FIG. 8A

SUBSTITUTE SHEET (RULE 26)